UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Automatic Material Recognition Using Computer Vision

Trabajo Final de Grado

Grado en Ingeniería Informática

UIUC advisor
David A. Forsyth
<daf@illinois.edu>

UPV advisor
Roberto Paredes
<rparedes@dsic.upv.es>

Student
José Vicente Ruiz Cepeda
<josruice@upv.es>

Curso 2013 / 2014

Resumen

La habilidad de identificar los materiales de los objetos del mundo real es clave para los seres humanos. Por ello, un sistema automático con esta capacidad sería un gran paso en el camino hacia la simulación de la visión humana. Este trabajo describe una nueva aproximación para intentar resolver este problema aún abierto, donde la idea clave es el uso de propiedades inferibles a través de las imágenes como paso intermedio para reconocer materiales.

El algoritmo se basa en la extracción y cuantización de descriptores SIFT*, que son usados para entrenar* SVMs *en el reconocimiento de propiedades de los materiales, tanto táctiles como de forma, en diferentes escalas. Finalmente, modelos basados en* Naive Bayes *y* SVMs *son entrenados y comparados para clasificar materiales a partir de las propiedades. A pesar del alto costo del marcado, esta aproximación ofrece interesantes resultados intermedios y precisión competitiva con otros métodos utilizados sobre el mismo conjunto de imágenes.*

Palabras clave: Vision por Computador, reconocimiento de materiales, SVM, SIFT.

Abstract

The ability to correctly identify the materials of objects in the real world is key in human's life. Therefore, an automatic system with that skill would be a great step in the path to simulate human vision. This work describes a new approach to try to solve this still open problem where the key idea is the use of image-inferable properties as an intermediate step to recognize materials.

The algorithm is based on the extraction and quantization of SIFT *descriptors, which are later used to train* SVMs *to recognize material properties, both of touch and shape, at different scales. Finally, models based on* Naive Bayes *and* SVMs *are trained and compared in terms of material recognition from the properties. Despite of the high cost of the markup, this approach offers interesting intermediate results and competitive accuracy compared with other methods applied to the same image dataset.*

Keywords: Computer Vision, material recognition, SVM, SIFT.

Contents

1 Introduction

The word *material* has its origin in the *Latin* word '*materia*', in English, *matter*. In the same way that reality is made of *matter*, every concrete entity can be defined in terms of materials: the silk of a dress, the scales of a python or the concrete of the pavement, for example. Therefore, materials represent an important part of the real world, since they are the key to understand and infer the physical properties of objects just looking at them.

Human beings, thanks to thousand of years of evolution, have become very good at recognizing materials. This skill could even determine the difference between life and death in the beginning of times, when being able to start a fire depended on choosing the right type of rock to make sparks. Nowadays, fortunately, our life does not depend on this ability, but we still use it everyday, even without noticing, for example, when deciding which side of a sidewalk step to avoid slippery ice during winter.

Because of all of this, an automatic system able to see the objects and recognize their materials would be really valuable, since it would be one more step on the way of building a machine able to see as well as a human, or even better. In that sense, despite of the achievements of *Computer Vision*, material recognition is still an open problem in the field.

We present a new approach in the problem of materials recognition, using feature extraction methods and *Support Vector Machines* (*SVMs*) to obtain the physical properties of the objects and applying *SVMs* again to this intermediate dataset to obtain the final material category in a set of 18 possible ones. The obtained results using the dataset of [3] beat any other system used on this dataset and prove that this approach has a great potential as a first step in the solution to the problem.

2 Dataset

The dataset comes from [3] and is composed of 18 material categories:

1. Birch.
2. Brick.
3. Concrete.
4. Corduroy.
5. Denim.
6. Elm.
7. Feathers.
8. Fur.
9. Hair.
10. KnitAran.
11. KnitGuernsey.
12. Leather.
13. Marble.
14. Scale.
15. Silk.
16. Slate.
17. Stucco.
18. Velour.

Each category contains 12 different sample images in `PNG` format with resolutions generally around 400 x 400 pixels. Therefore, the total number of sample images is 216, two of which can be seen in figures 1 and 2.

Figure 1: `Birch/04.png`

Figure 2: `Denim/07.png`

Each sample contains a material in the central part of the image that, most of the times, extends till the borders of it. In some cases, watermarks and other kinds of noise appear in the image. The whole dataset, with reduced scale and each category in one row, can be seen in figure 4.

Figure 3: Example of images with noise: `Hair/10.png` and `Knitguernsey/02.png`

Figure 4: Complete dataset

3 Algorithm

3.1 Outline

The classification process is divided in two main phases:

- In the first phase, the material properties are inferred from the images. To achieve this, *SIFT* descriptors are extracted from all the sample images. Next, a vector quantization process is applied over them using *k-means*. The results are finally used as input feature vectors for the *SVMs*, along with manually defined markup of properties.

- In the second phase, the material label is obtained using the inferred material properties. For this purpose, the vectors of properties and the real material labels are used as input for *SVMs* and *Naive Bayes* classifiers.

3.2 From images to properties

3.2.1 Manual markup of properties

The first step is to manually markup the properties of each of the sample images. This procedure consists on defining its shape and touch for the *fine*, *medium* and *coarse* scales. Figure 5 shows an example of the approximate size of them, although its position can be arbitrary.

Figure 5: Visual representation of the different scales.

The possible values assigned to the three scales for each of the properties are the following:

- **Shape**: *flat*, *round*, *extended organized* or *extended disorganized*.
- **Touch**: *furry*, *feathery*, *coarse*, *bumpy*, *scratchy*, *smooth* or *velvety*.

An example of a manually marked image is the following one:

Brick 02:

- **Fine**: flat, scratchy.
- **Medium**: extended organized, coarse.
- **Coarse**: extended organized, coarse.

In this case, the material shows a scratchy touch at fine scale, due to the mortar used to join the bricks. At this scale, the shape can be considered flat, since no forms are appreciable. At a medium scale, its touch is mainly coarse, because of the bricks, while some forms extended and organized start to appear. Both properties are the same if the image is considered at a coarse scale.

In many cases, several touch or shape properties are available at a given scale in the same sample image. No precedence rules have been defined at this point, which means that any of them is completely valid.

3.2.2 Extraction of Scale-Invariant Feature Transform (SIFT) descriptors

The next step of the*Automatic Material Recognition* algorithm is the extraction of *Scale-Invariant Feature Transform* (*SIFT*) descriptors from each of the sample images.

In 2004, David Lowe, from the *University of British Columbia*, came up with this new algorithm in [5] to extract keypoints and compute its descriptors from an image. The algorithm involves the following main steps:

1. First, potential keypoints are detected. For this, a *Difference of Gaussian* (*DoG*), is convoluted using various values of σ, which acts as a scaling parameter, with the gray-scale version of the input image. Then, the potential keypoints are obtained finding the local extrema over scale and space in the *DoG*.

2. Next, once potential keypoints are found, they are filtered to maintain only the interesting ones and increase the accuracy of the results. To achieve this, keypoints containing edges and those without enough intensity are discarded.

3. Then, an orientation is assigned to each keypoint to achieve invariance to image rotation. For this, the gradient magnitude and direction is calculated in the neighbourhood of each keypoint.

4. Finally, keypoint descriptors are created. For this, a 4x4 neighbourhood is takend around the keypoint and a 8 bin orientation histogram is created for each sub-block. This adds up to a total of 128 bin values available, which are represented as a vector to form the keypoint descriptor.

A visual example of the keypoints and descriptors extracted from the material images using *SIFT* algorithm can be seen in figures 6, 7, 8 and 9.

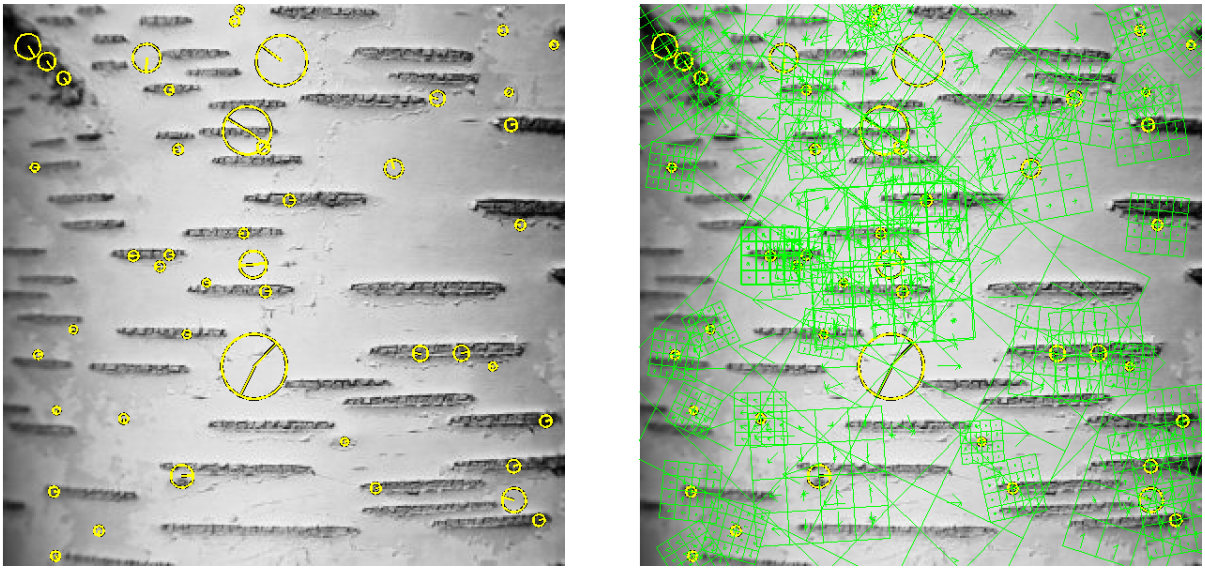


Figure 6: 50 randomly sampled keypoints (on the left) and keypoints with descriptors (on the right) extracted with *SIFT* from `Birch/10.png`.

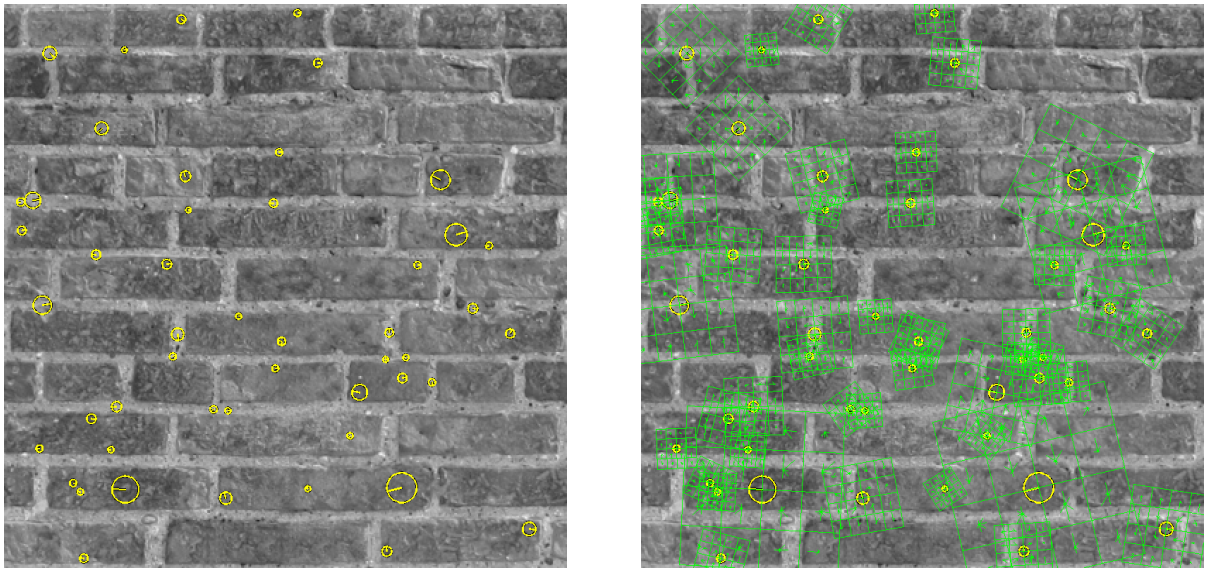


Figure 7: 50 randomly sampled keypoints (on the left) and keypoints with descriptors (on the right) extracted with *SIFT* from `Brick/02.png`.

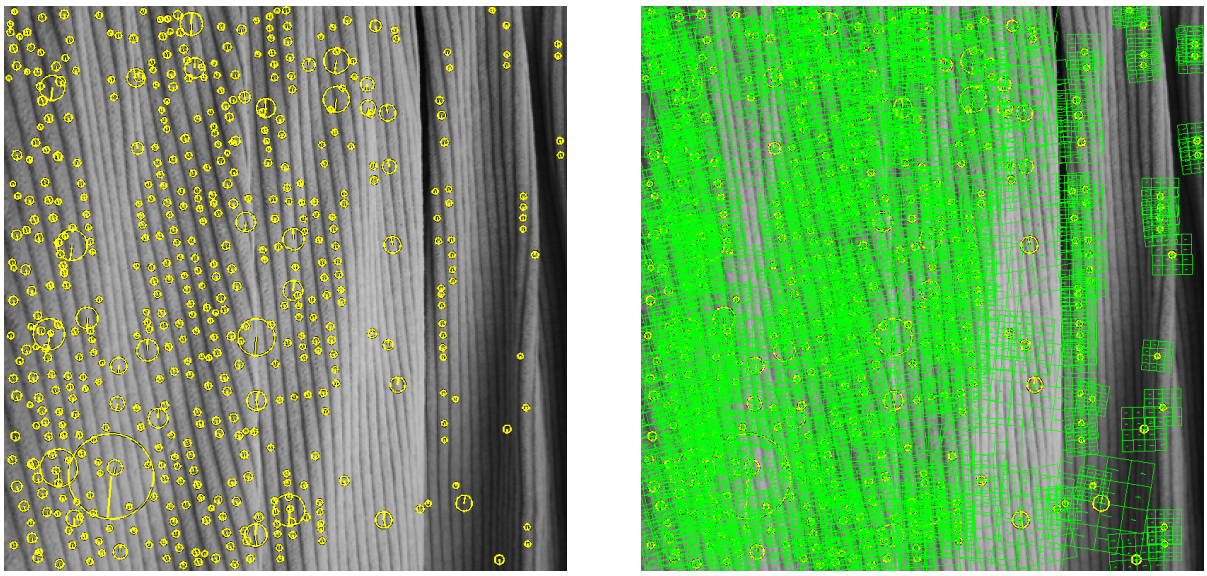


Figure 8: All the keypoints (on the left) and keypoints with descriptors (on the right) extracted with *SIFT* from `Corduroy/05.png`.

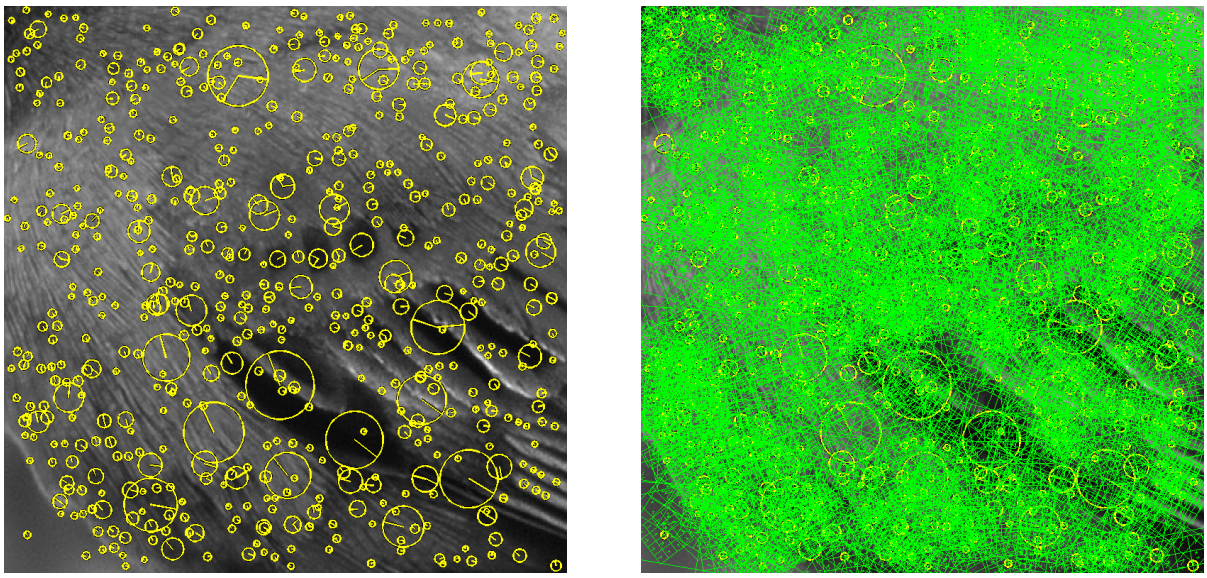


Figure 9: All the keypoints (on the left) and keypoints with descriptors (on the right) extracted with *SIFT* from `Feathers/11.png`.

Apart from the standard *SIFT* implementation, a dense version of it, called *PHOW*, have been used. The *MATLAB* code where this step is implemented and commented at a low-level can be found in the file `computeDescriptors.m`.

3.2.3 Vector quantization through *k-means* clustering

After all the *SIFT* descriptors have been extracted from the material images, a vector quantization process is required, since the raw descriptors cannot be used directly as input features for the *SVMs*. This is due to the variability in the number of descriptors computed for each image. Even if that were not the case, a preprocess step would still be required to convert all the 128-dimensional descriptors of every image into a suitable feature vector.

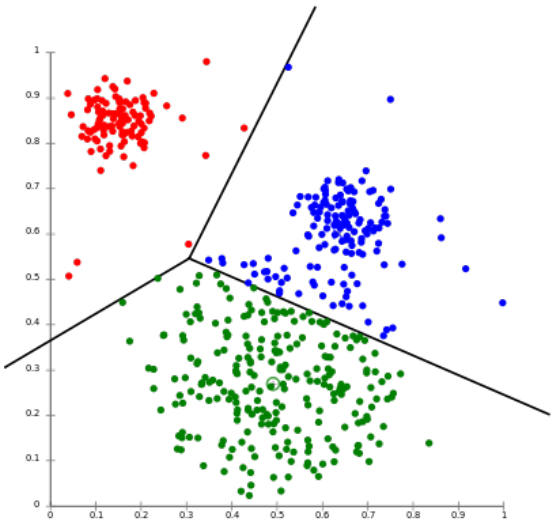


Figure 10: A visual representation[1] of the result of *k-means* applied to two-dimensional observations for $k = 3$ cluster centers.

The vector quantization is based on a *bag-of-words model* to generate the feature vector of each image. First, *k-means* clustering is applied to the extracted *SIFT* descriptors.

K-means clustering is an algorithm original from signal processing and proposed by Stuart Lloyd in 1982 in its standard version as a technique for *pulse-code modulation* in [4]. It aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. To achieve this, it starts selecting k random observations and setting them as cluster centers. Then, it assigns each observation to the closest cluster center and updates the cluster centers by computing the mean of the assigned observations. It repeats the same process till convergence.

Therefore, the *SIFT* descriptors are partitioned into k clusters, generating k clusters centers, which work as *codewords* in the *bag-of-words model*. Finally, each image is assigned a histogram of the *codewords*, which will work as feature vectors in the *SVMs*, by counting the number of *SIFT* descriptors of that image that are "represented" by each *codeword*.

The *MATLAB* code where this step is implemented and commented at a low-level can be found in the file `quantizeVectors.m`.

[1] *"KMeans-Gaussian-data"* by Chire - Own work. Licensed under Creative Commons Attribution-Share Alike 3.0 via Wikimedia Commons.

3.2.4 Support Vector Machines (SVMs) for properties recognition

Once every sample image has its own feature vectors associated, resulting from the vector quantization of the previous step, it is time to train a classifier using them along with the manual markup of properties, which will be the class labels that the classifier will produce. For this purpose, *Support Vector Machines* (*SMVs*) have been used.

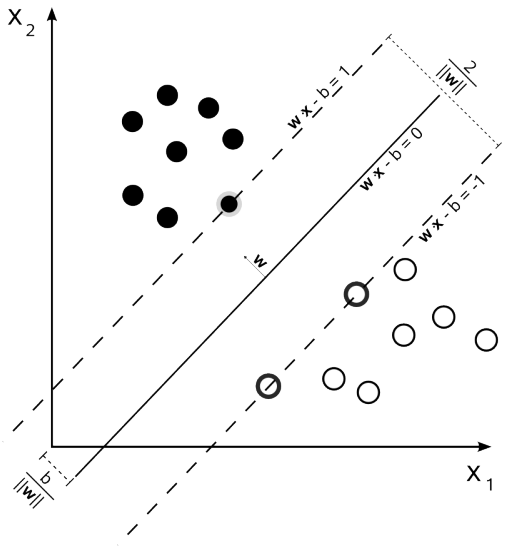


Figure 11: A visual representation[2] of the result of the model obtained by a *SVM* algorithm to two-dimensional feature vectors.

SVMs, invented by Vladimir Vapnik and published for the first time in [1], are supervised learning models that work as binary linear classifiers. Given a set of training examples with their respective binary class labels, an *SVM* training algorithm builds a model that assigns, i.e. classifies, new examples into one of the two categories. The built model will define a $(k-1)$-dimensional hyperplane (where k is the number of clusters used in the previous step) that will represent the largest separation between the two classes.

In our case, one *SVM* is trained for each possible combination of properties and scales that exist in, at least, one of the images of the dataset. In other words, one of the *SVMs* will be used to tell apart material images that present the touch property *bumpy* at *medium* scale, and a different one will decide if a sample is *bumpy* at *coarse* scale, or *extended-organized* at *fine* scale, etc. In the same way, if no image has been manually classified as *feathery* at *fine* scale, no *SVM* will be trained for that property, because of the lack of positive samples. Therefore, the resulting *SVMs*, individually, will only be able to tell if a sample image presents the property at a specific scale or not, since they are binary classifiers.

The *MATLAB* code where this step is implemented and commented at a low-level can be found in the file `svm.m`.

3.3 From properties to materials

After the properties have been inferred from the material images, a binary feature vector can be built from them for each image considering all the *property-scale* combinations and assigning a 1 if the image has that property or a 0 otherwise.

3.3.1 Support Vector Machines (SVMs) for material recognition

The feature vectors combined with the real material labels, allow us to train new *SVMs* for the task of classifying materials from the properties. Unfortunately, the *SVM* model is only suitable for binary classification, i.e., if an image contains a material or not, which is not enough for

[2] *"Svm max sep hyperplane with margin"* by Cyc - Own work. Licensed under Public domain via Wikimedia Commons.

multi-class classification problem like this.

To solve this problem, *Machine Learning* theorists have come up with different strategies. One of them, and the one we use, is *One Vs. All* (*OvA*, a.k.a. *One Vs. Rest* or *OvR*), which consists on training a binary classifier por each class and, when classifying a new sample, applying all of them individually over the sample and considering as result the class of the classifier that gets the highest confidence in the classification. So, for example, if our problem considered three possible materials: *marble*, *brick* and *leather*, and the results of their respective *SVMs* were 3.52, 6.07 and -1.89, the *OvA* would determine that the correct class is *brick*.

The *MATLAB* code where this step is implemented and commented at a low-level can be found in the file `svmOneVsAll.m`.

3.3.2 Naive Bayes classifier for material recognition

The properties feature vectors can also be used to fed a different kind of classifier, in this case, more suitable for multi-class classification, as is a *Naive Bayes* classifier.

Naive Bayes classifiers are another kind of supervised learning models which, unlike *SVMs*, work as probabilistic classifiers instead of linear ones. They are based on the application of the *Bayes' theorem* with strong, i.e. naive, independence assumptions between the features. In other words, it is assummed that the features are independent with respect to each other given the class.

With all the information computed in the previous steps about the images, a *Naive Bayes* classifier can be trained for material recognition. The *MATLAB* code where this step is implemented and commented at a low-level can be found in the file `naiveBayes.m`.

4 Approach motivation

The novelty of the approach strives in the intermediate step of obtaining the material properties in the process of recognition. In other words, instead of trying to infer the class directly from the image descriptors, the properties of the materials are obtained first, and from them, the class of the material. Therefore, its effectiveness depends on the following:

- The image feature descriptors, along with the quantization process, are powerful enough to capture precisely the subtleties that determine if a material image presents certain property at a specific scale.

- The vocabulary of properties and scales is sufficiently wide and varied to allow a clear distinction between materials.

The ability to choose the most suitable techniques to address this issues will determine the accuracy of the approach.

4.1 Advantages

Thanks to the intermediate classification step, the approach presents several clear advantages:

- In case of absolute lack of samples of one class, the approach would, at least, be able to compute its properties if they are present in other samples of the dataset.
- A classification of an image with only one training sample of that same material would be possible with relative robustness, again, if its properties are present in other samples.

4.2 Drawbacks

Also, some drawbacks are inherent to the use of more steps in the classification process:

- The recognition of material properties is based on a supervised learning that requires a costly manual markup of the properties. Unfortunately, the more powerful, varied and precise the vocabulary, the more costly the process of marking the images.
- The computational cost of the algorithm increases because of the double recognition problem, first properties, then materials.

5 Engineering

5.1 Tools

The main tool used while completing this project has been *MATLAB*. Its powerful built-in functions and routines for vector and matrix manipulation, along with its wide popularity in academia and, in particular, among the image processing scientists, made it really suitable for the purpose. All the code has been written in *MATLAB* programming language and the debugger that comes with its *IDE* has been widely used to test and polish it.

The text editor used to write all the code has been *Sublime Text 2*, because of its flexibility, *MATLAB* syntax highlight and features like multiple cursors or quick selection of the same word, which make the coding task much easier and smooth.

This report has been generated using LaTeX. The main reason to choose it over a *WYSIWYG* editor are the look and feel of the result and its flexibility in many aspects, like the definition of reusable functions or the availability of packages for almost any imaginable task.

5.2 Libraries

VLFeat has been the main library for this project. Its functions for *SIFT*, *k-means* and *SVMs* have been really helpful for the implementation. For a complete reference about the library, [6] should be checked.

5.3 Maintenance

Every part of this project has been included in the version-control system *Git* since the beginning, and uploaded frequently to *Github*[3], a website where the code is publicly available.

Besides, *MATLAB* code has been written taking into account one of the most important styleguides of the language [2], to produce code cleaner and easier to read and understand. Also, comments have been included to emphasize even more the later point.

6 Results

6.1 Parameter tuning

Given that several steps of the algorithm produce slightly different outputs depending on some predefined parameters, it is natural to have an interest to find the combination of parameter values that gives the best results.

In order to achieve that, the following parameters were considered:

- Maximum number of *SIFT* descriptors per sample image and, related to it, density of *SIFT* while computing them. If the amount of descriptors exceeded the maximum allowed, they were randomly sampled. Its default value was non-dense *SIFT* without maximum amount of descriptors limitation. The results are shown in figures 12 and 13.
- k parameter in *k-means*, i.e. number of clusters, with a default value of 300. Its results can be seen in figures 16 and 17.
- λ value in *SVMs*, a regularization parameter directly related with the number of iterations performed while computing the best separation hyperplane. Its default value was 10^{-5}, and the results are represented by figures 14 and 15.

The testing procedure consisted on applying *Cross-Validation* to the dataset, using as test samples a set of 18 images, one per material class, each time. With respect to *Cross-Validation*, it is a widely used model validation technique that allows to measure how the results will generalize to an independent dataset.

As can be seen, the models trained using as feature vectors the material properties from the markup get a really good accuracy, around 80%, almost invariably. This invariability makes sense, specially in the case of the parameter tuning for the number of clusters and the *SIFT* descriptors, since the model are not affected by those.

On the other side, the models trained with predicted feature vectors, obtain accuracies around 30% - 40% in all cases. When tested independently, the best value of each parameter is dense *SIFT* (*PHOW*) with a maximum of 1000 descriptors per image, 800 clusters for *k-means* and a *SVMs* $\lambda = 0.01$. Nevertheless, when tested as a whole, the best accuracy was obtained by a different, although close, set of parameters, as will be seen.

[3] `https://github.com/josruice/CS499`

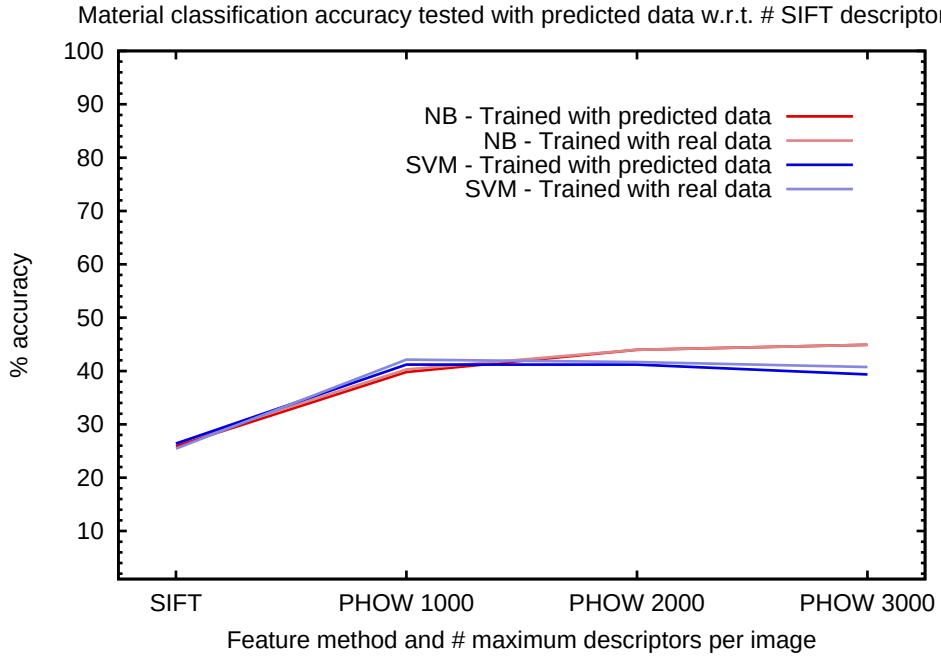


Figure 12: Graph showing the relation between the material recognition accuracy and the maximum number of *SIFT* descriptors per image, using 300 clusters and *SVMs* $\lambda = 10^{-5}$. All these results were obtained using predicted data, i.e. the predicted material properties. In the case of the x-label *SIFT*, no maximum was forced.

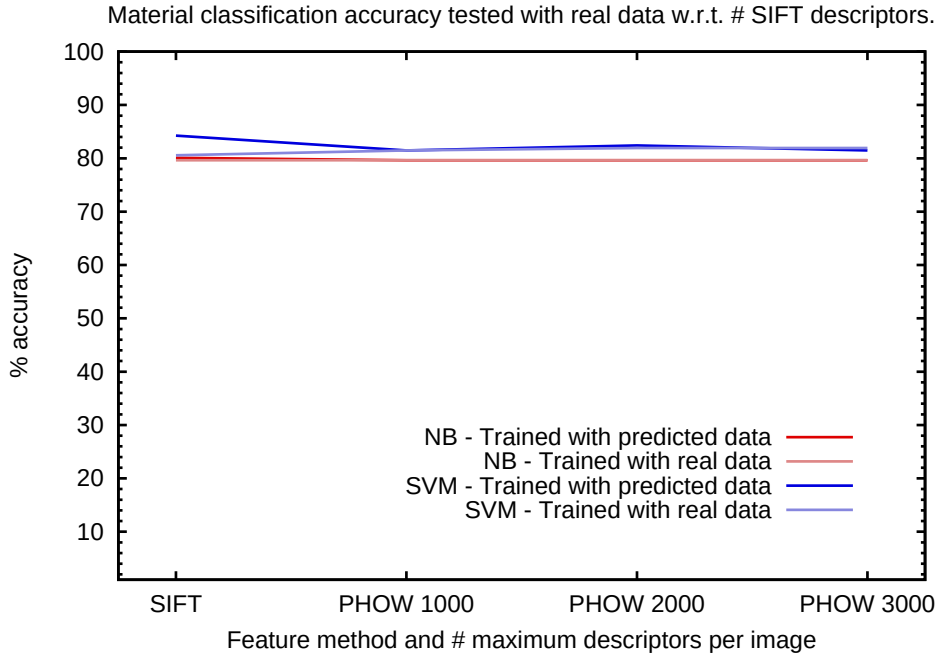


Figure 13: Graph showing the relation between the material recognition accuracy and the maximum number of *SIFT* descriptors per image, using 300 clusters and *SVMs* $\lambda = 10^{-5}$. All these results were obtained using real data, i.e. the material properties from the manual markup. In the case of the x-label *SIFT*, no maximum was forced.

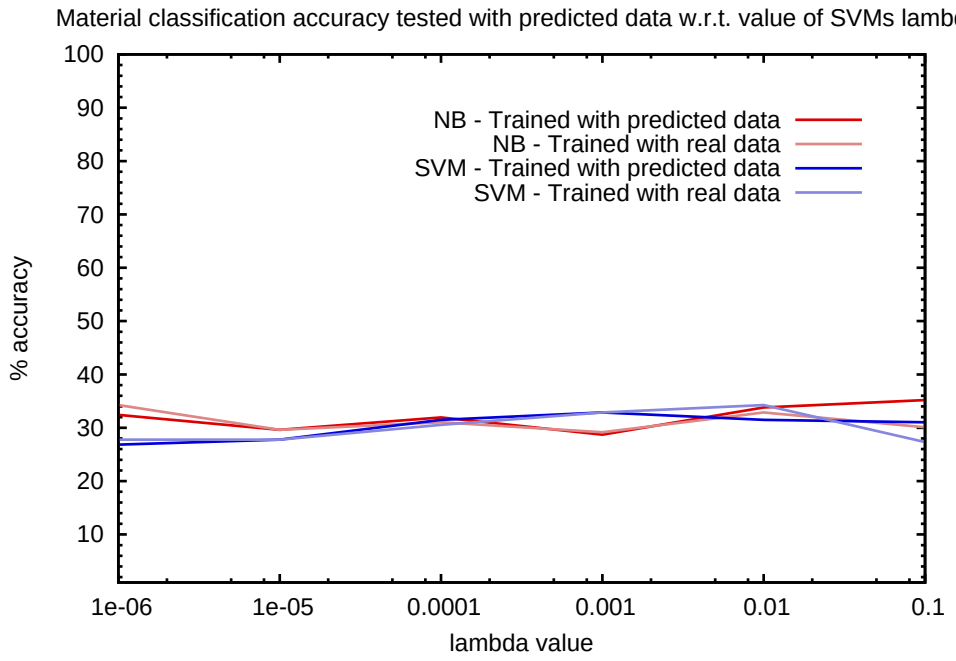


Figure 14: Graph showing the relation between the material recognition accuracy and the value of *SVMs* λ, using 300 clusters and non-dense *SIFT* without maximum descriptors limitation. All these results were obtained using predicted data, i.e. the predicted material properties.

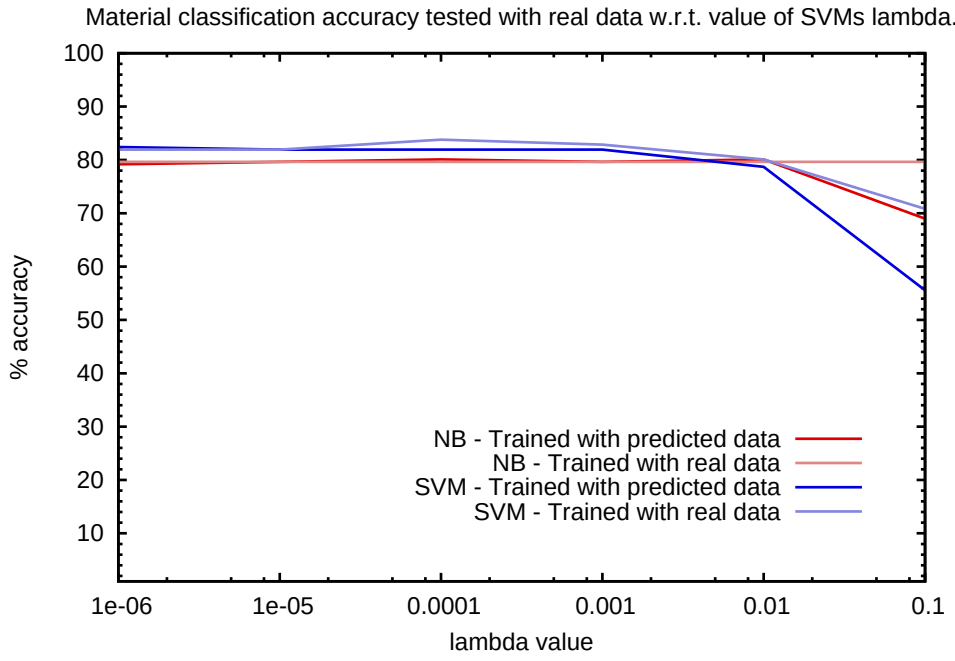


Figure 15: Graph showing the relation between the material recognition accuracy and the value of *SVMs* λ, using 300 clusters and non-dense *SIFT* without maximum descriptors limitation. All these results were obtained using real data, i.e. the material properties from the manual markup.

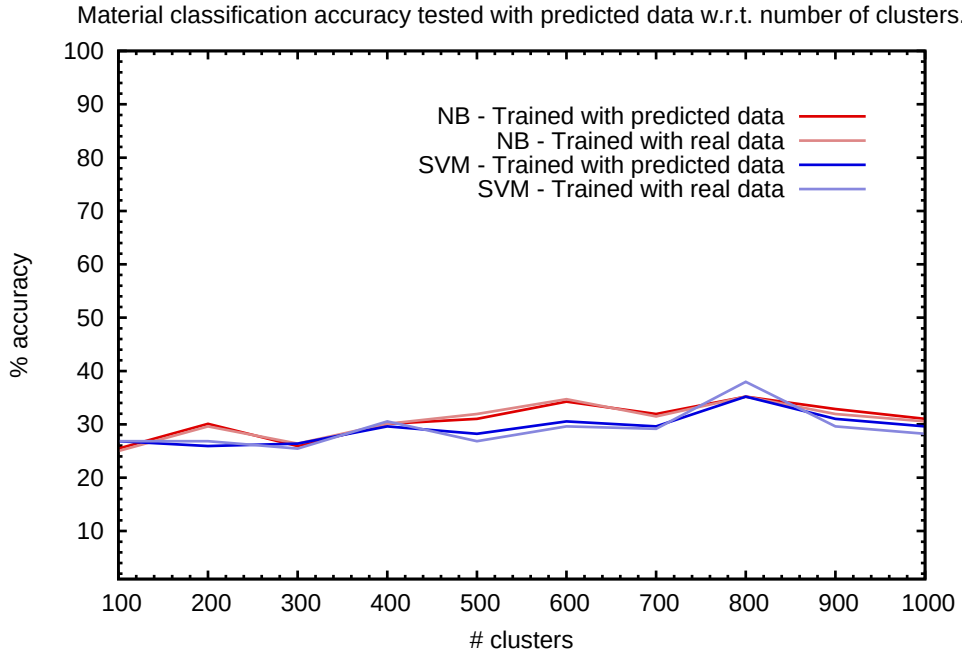


Figure 16: Graph showing the relation between the material recognition accuracy and the number of clusters in *k-means*, using *SVMs* $\lambda = 10^{-5}$ and non-dense *SIFT* without maximum descriptors limitation. All these results were obtained using predicted data, i.e. the predicted material properties.

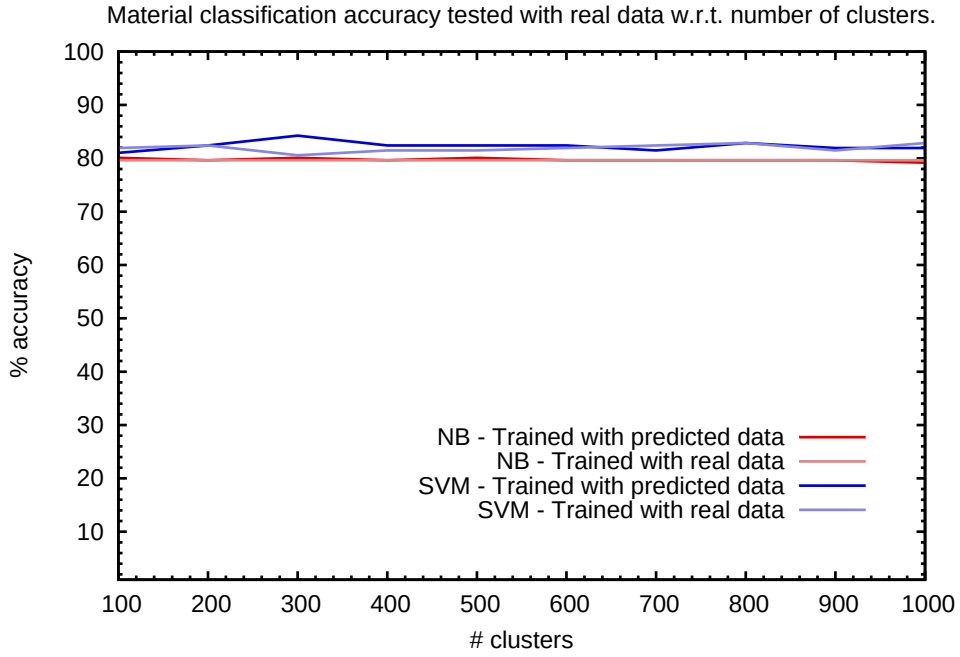


Figure 17: Graph showing the relation between the material recognition accuracy and the number of clusters in *k-means*, using *SVMs* $\lambda = 10^{-5}$ and non-dense *SIFT* without maximum descriptors limitation. All these results were obtained using real data, i.e. the material properties from the manual markup.

6.2 Properties recognition accuracy

As previously said, one of the most important parts of the algorithm is its ability to correctly identify the properties present on the material images. To measure this, the following three metrics have been considered:

- **Global accuracy**: Percentage of classifications where the algorithm correctly identifies that a property is present or that is not present. In other words, true positives plus true negatives, all divided by the total number of samples.
- **Precision**: Percentage of classifications where the algorithm correctly identifies that a property is present over the total number of times it identifies (correctly or uncorrectly) that same property is present, i.e., true positives divided by true positives plus false positives.
- **Recall**: Percentage of classifications where the algorithm correctly identifies that a property is present over the total number of samples that actually present that property. Technically, true positives divided by true positives plus true negatives.

The results have been obtained using *Cross-validation* with 12 partitions, guaranteeing that exactly one image of each class was present in every test set. The bar charts for the two first iterations have been plotted, with two versions for every previous metrics: one sorted by material property name (x-axis) and another sorted by the metric value itself (y-axis). The bar charts for the first partition can be seen in the figures 18, 19, 20, 21, 22 and 23, while the ones for the second partition are represented in the figures 24, 25, 26, 27, 28 and 29.

A careful look at the results, reveals that, although the global accuracy seems pretty high for most properties, there is a great room for improvement, as the precision and recall show. Furthermore, in the case of the global accurary, a random classifiers would yield a value approximate to 50%, which is close, even from the upper part, to the performance obtained in some properties.

It can be seen how some properties, like a *flat* shape at *fine* scale, or a *extended disorganized* shape at *medium* scale are hard to recognize for the current set up. On the other side, several touch properties are well recognized at different scales, achieving even a 100% accuracy.

The conclusion that can be extracted from these results is that while *SIFT* descriptors plus vector quantization technique is able to capture enough details for the touch properties, there is still room for improvement in the recognition of shapes.

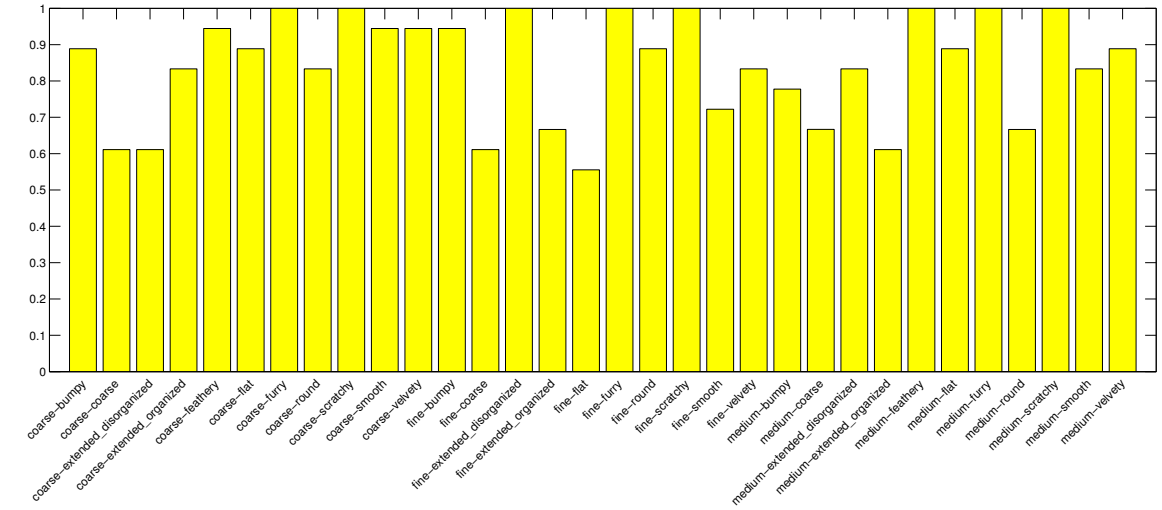


Figure 18: Bar chart showing the global accuracy in the detection of a property or the lack of it, sorted by *scale-property*. The test set was composed of the first image of each material class, while the rest of the images were part of the training set. Measurements obtained using dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$.

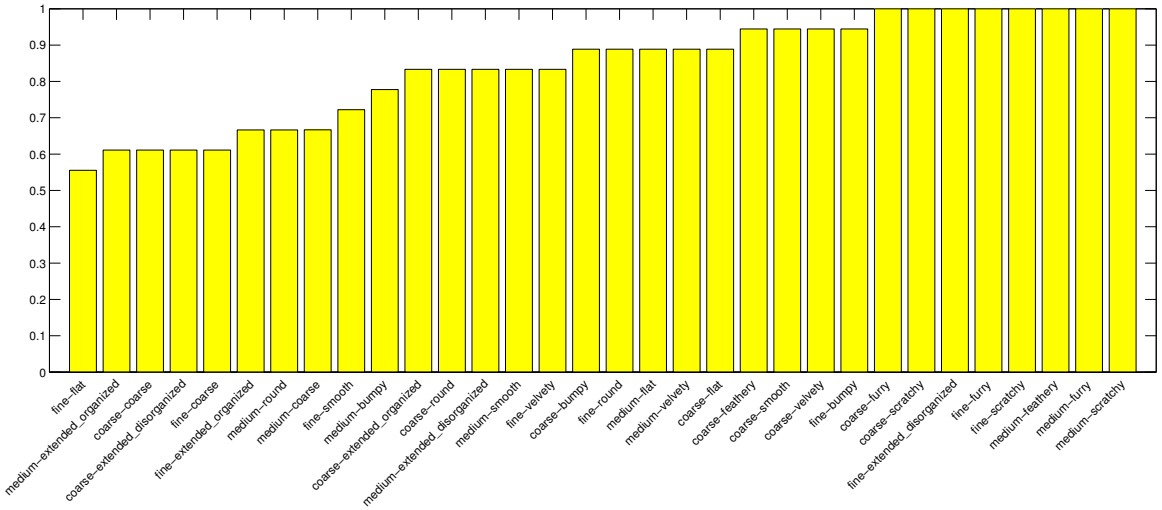


Figure 19: Bar chart showing the global accuracy in the detection of a property or the lack of it, sorted by the accuracy itself. The test set was composed of the first image of each material class, while the rest of the images were part of the training set. Measurements obtained using dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$.

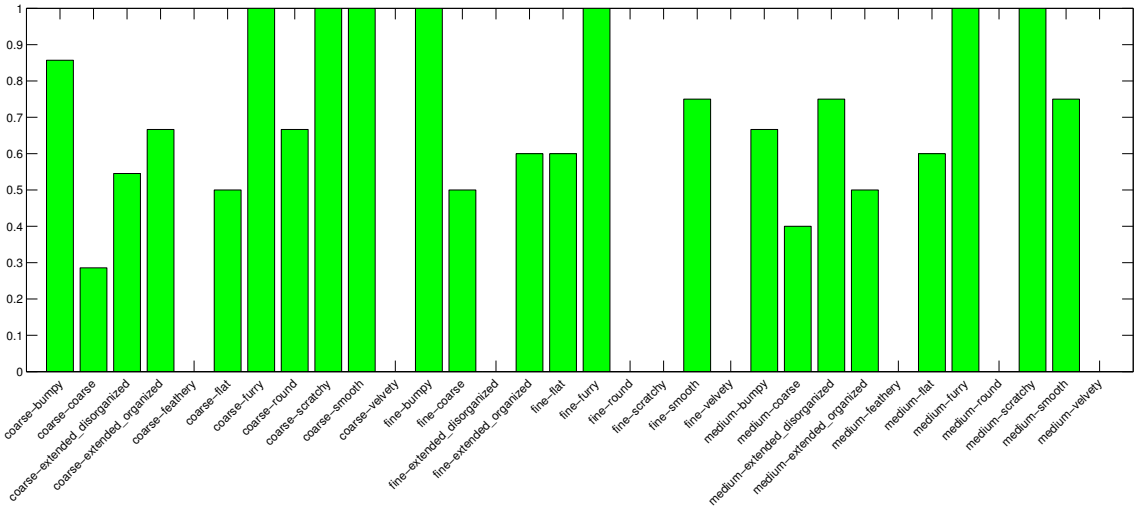


Figure 20: Bar chart showing the *precision* in the detection of a property, i.e. the number of true positives over the total number of image samples that has been classified as having the property, sorted by *scale-property*. The test set was composed of the first image of each material class, while the rest of the images were part of the training set. Measurements obtained using dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$.

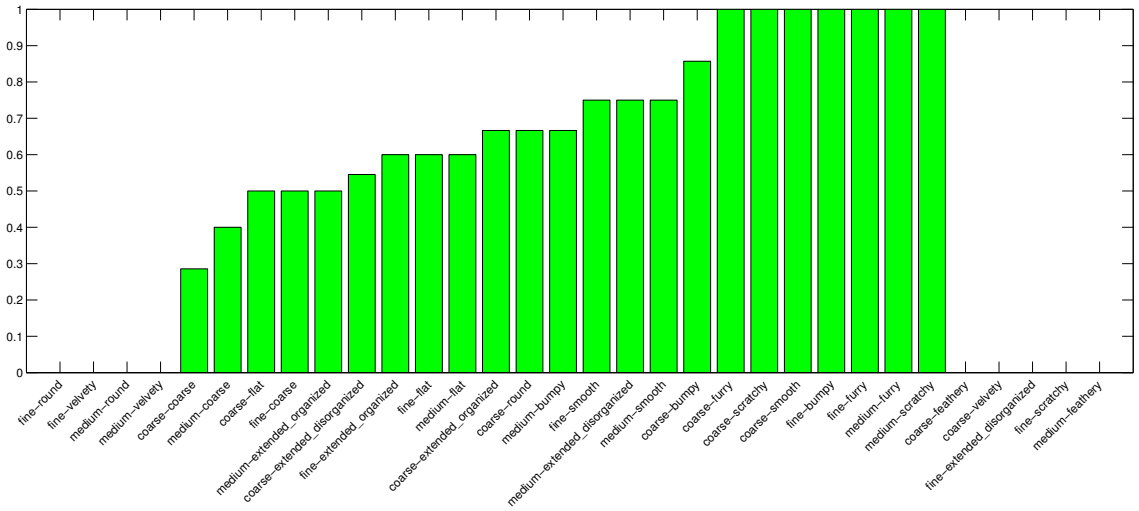


Figure 21: Bar chart showing the *precision* in the detection of a property, i.e. the number of true positives over the total number of image samples that has been classified as having the property, sorted by the *precision* itself. The test set was composed of the first image of each material class, while the rest of the images were part of the training set. Measurements obtained using dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$.

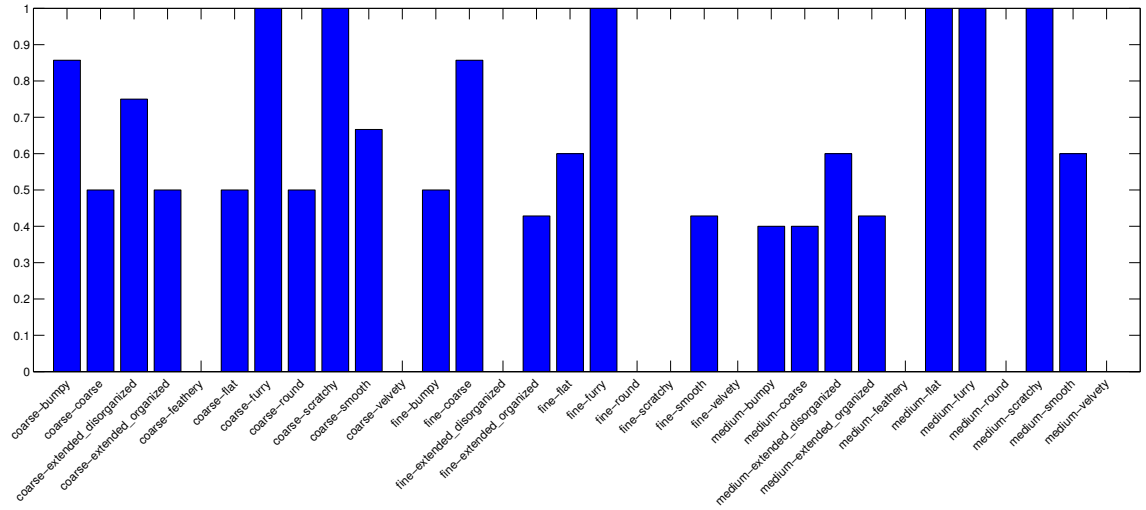


Figure 22: Bar chart showing the *recall* in the detection of a property, i.e. the number of true positives over the total number of image samples with the property, sorted by *scale-property*. The test set was composed of the first image of each material class, while the rest of the images were part of the training set. Measurements obtained using dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$.

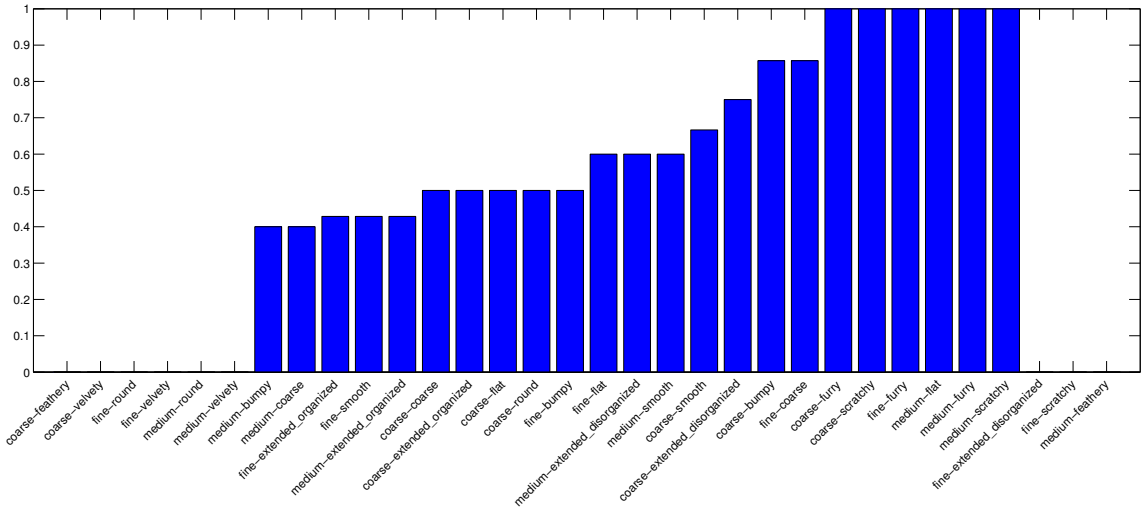


Figure 23: Bar chart showing the *recall* in the detection of a property, i.e. the number of true positives over the total number of image samples with the property, sorted by the *recall* itself. The test set was composed of the first image of each material class, while the rest of the images were part of the training set. Measurements obtained using dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$.

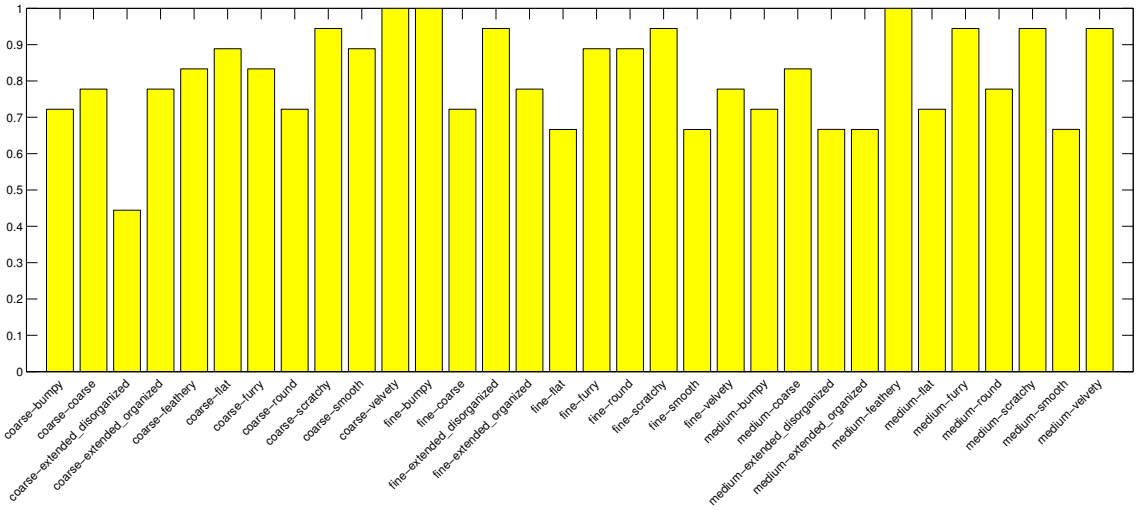


Figure 24: Bar chart showing the global accuracy in the detection of a property or the lack of ifig:t, sorted by *scale-property*. The test set was composed of the second image of each material class, while the rest of the images were part of the training set. Measurements obtained using dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$.

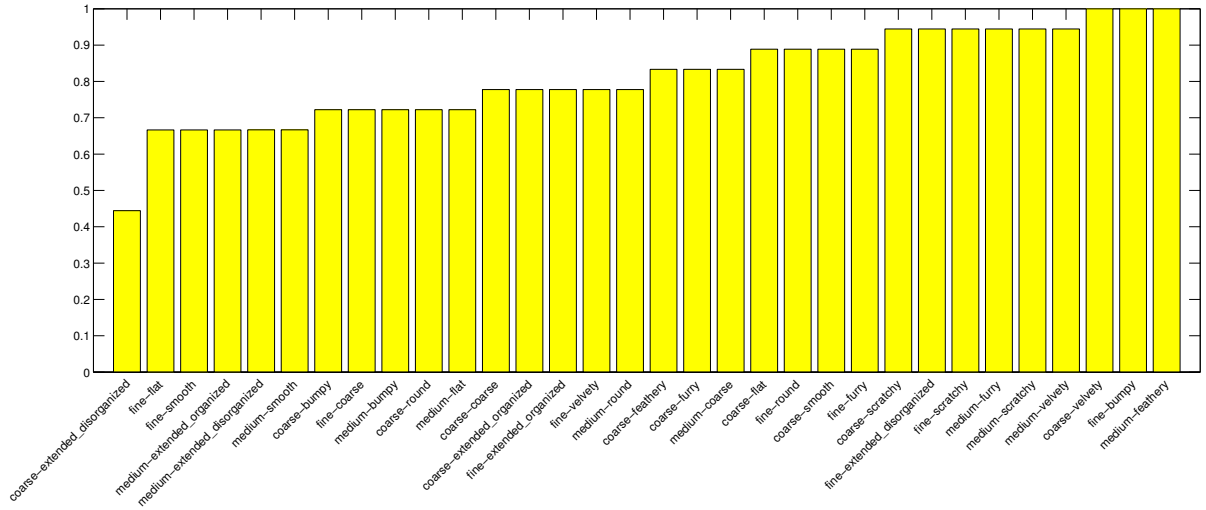


Figure 25: Bar chart showing the global accuracy in the detection of a property or the lack of ifig:t, sorted by the accuracy itself. The test set was composed of the second image of each material class, while the rest of the images were part of the training set. Measurements obtained using dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$.

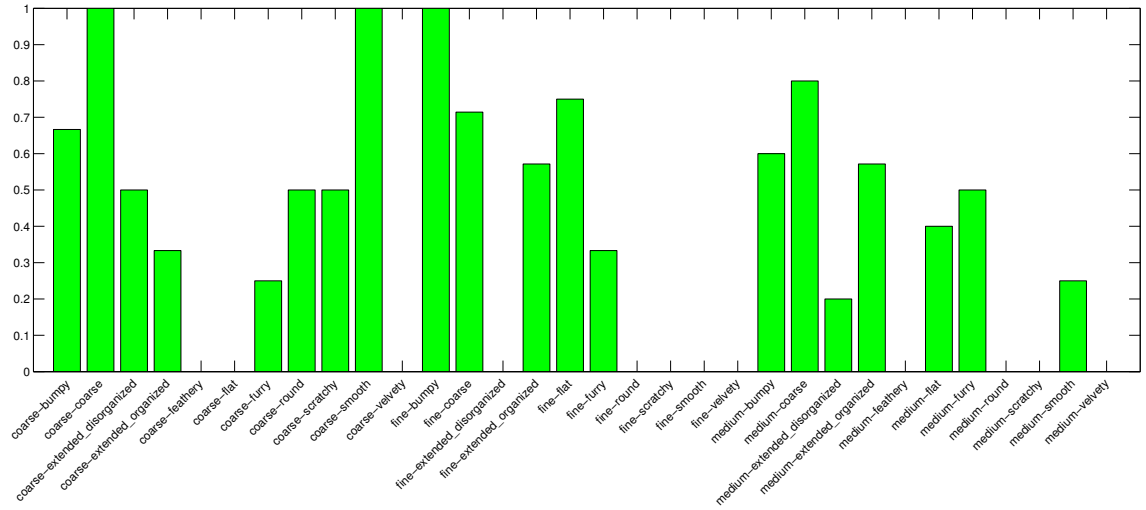


Figure 26: Bar chart showing the *precision* in the detection of a property, i.e. the number of true positives over the total number of image samples that has been classified as having the property, sorted by *scale-property*. The test set was composed of the second image of each material class, while the rest of the images were part of the training set. Measurements obtained using dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$.

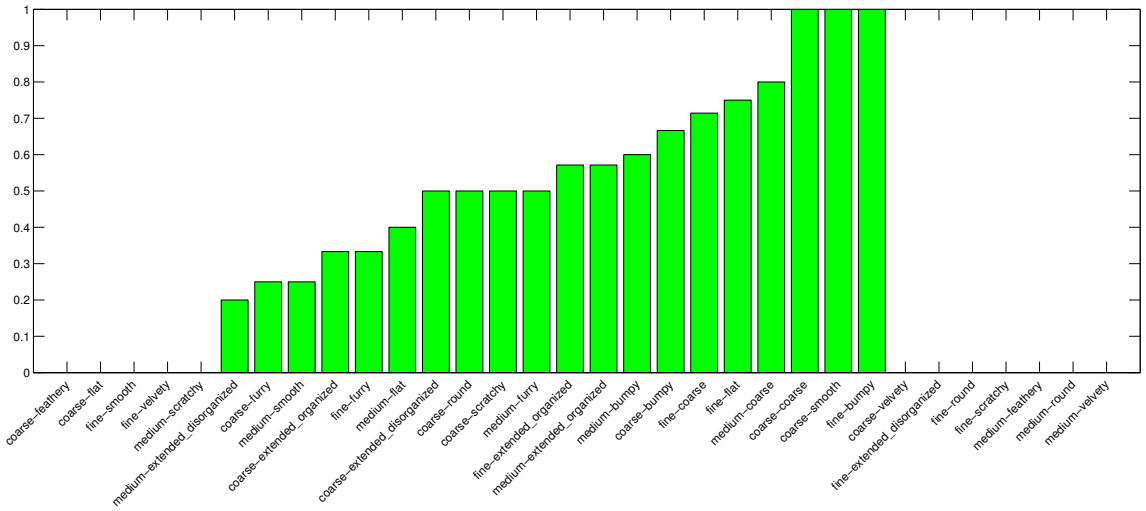


Figure 27: Bar chart showing the *precision* in the detection of a property, i.e. the number of true positives over the total number of image samples that has been classified as having the property, sorted by the *precision* itself. The test set was composed of the second image of each material class, while the rest of the images were part of the training set. Measurements obtained using dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$.

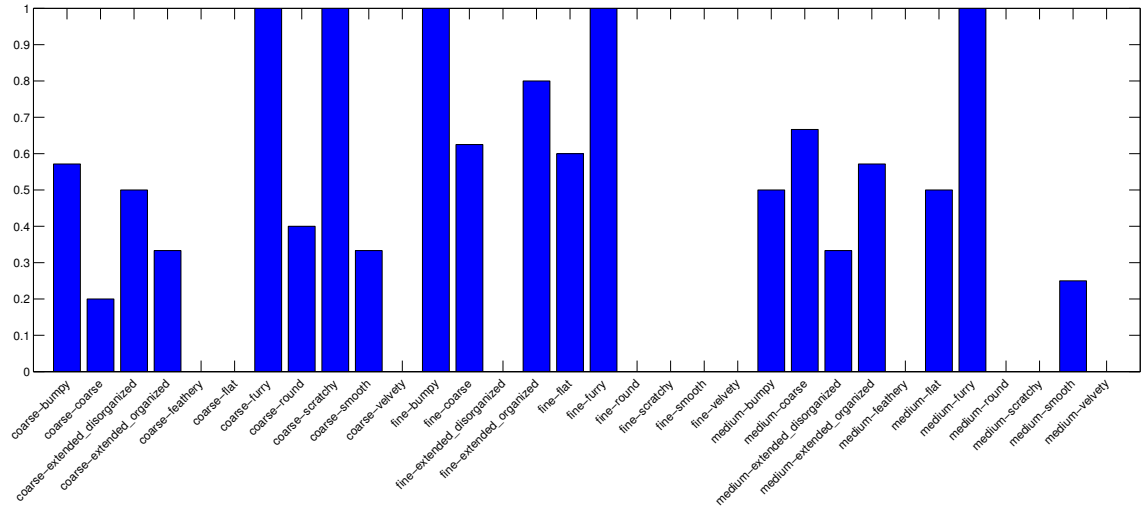


Figure 28: Bar chart showing the *recall* in the detection of a property, i.e. the number of true positives over the total number of image samples with the property, sorted by *scale-property*. The test set was composed of the second image of each material class, while the rest of the images were part of the training set. Measurements obtained using dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$.

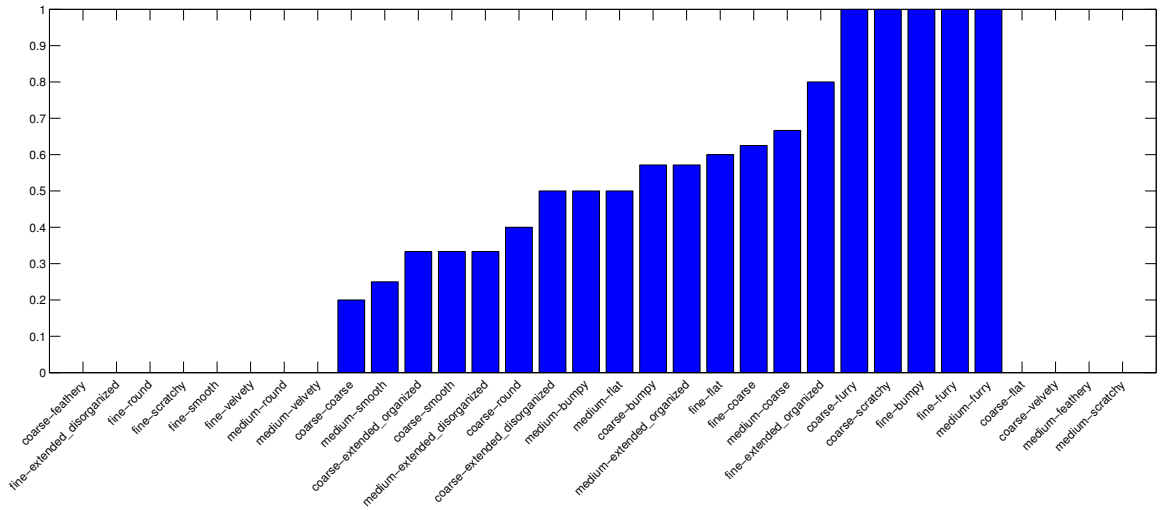


Figure 29: Bar chart showing the *recall* in the detection of a property, i.e. the number of true positives over the total number of image samples with the property, sorted by the *recall* itself. The test set was composed of the second image of each material class, while the rest of the images were part of the training set. Measurements obtained using dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$.

6.3 Confusion matrices

The last part of the algorithm consists on recognizing the material using the properties extracted from each image. There is no need to highlight the great importance of this step, since this is the main purpose for the whole algorithm.

A widely used method to measure the performance of multi-class classification algorithms are the *confusion matrices*. This mathematical tool represents the results as a matrix with the same number of rows and columns as possible classes, and in which the cell in row i, column j represents the percentage of times an image of i^{th} class has been classified as an image of j^{th} class. In this situation, the best classifiers will be those that achieve results closer to 1 (or 100%) in the main diagonal of the matrix, being the ideal a perfect diagonal matrix with all 1.

The *confusion matrices* represented in figures 34, 35, 36, 37, 30, 31, 32, 33, show the results for *SVMs* and *Naive Bayes* for all the possible combinations of training and testing using data predicted by the algorithm or data extracted from the manual markup, assummed to be real data.

As before, a *Cross-validation* process with 12 partitions have been used, guaranteeing that exactly one image of each class was present in every test set. In this case, the averaged sum of the confusion matrices of every iteration is represented in each image.

As can be seen, the most striking fact is that the classifiers tested with feature vectors extracted from the manual markup are able to classify with much higher accuracy, in some cases, almost as twice. The use of this kind of data for the testing part allow us to see the theoretical limitation of the model, since it represents the performance assumming perfect results in the process of recognizing the material properties. The fact that the achieved accuracy with this data is above 80% is a good sign of the ability of the vocabulary to capture the difference between material classes.

The *confusion matrices* also give us insights about what classes are close to each other in terms of properties, taking into account the current vocabulary. For example, a fair amount of *concrete* samples are classified as *stucco*, and the other way around. Also, it can be extracted which classes lose more information in the property recognition process, since they get almost no correct classifications when testing the predicted properties. Examples of this are *denim* and *knitguernsey*.

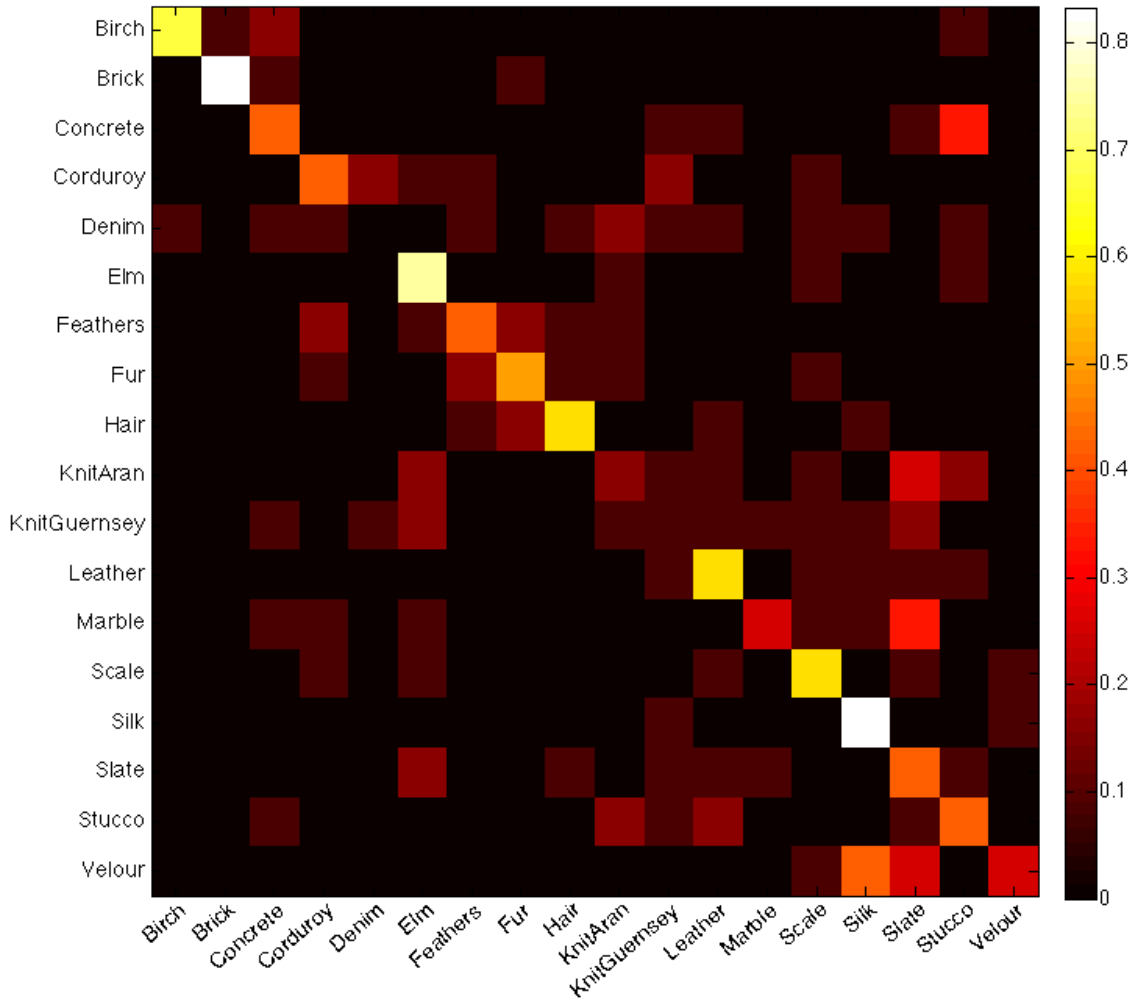


Figure 30: Heat map representing the confusion matrix of the material recognition process using *Naive Bayes* trained with predicted data, i.e., properties estimated algorithmically, and tested also with predicted data. The results are the average of a *cross-validation* process of 12 partitions, where the test set had at least one sample of each class. The parameters used are dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$. The average precision obtained is 45.37 %, with 98 out of 216 samples correctly classified.

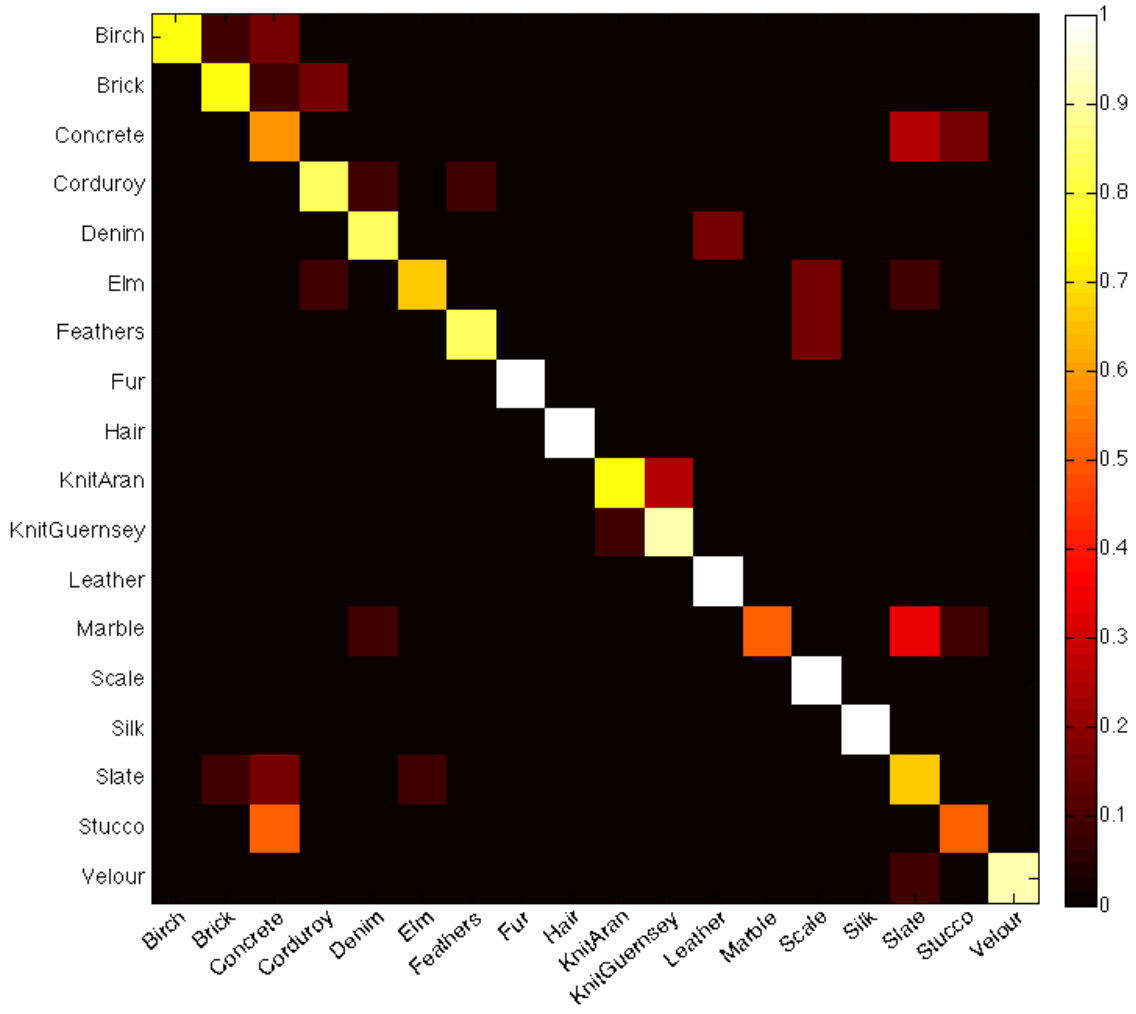


Figure 31: Heat map representing the confusion matrix of the material recognition process using *Naive Bayes* trained with predicted data, i.e., properties estimated algorithmically, and tested with real data, i.e., properties extracted from the manual markup. The results are the average of a *cross-validation* process of 12 partitions, where the test set had at least one sample of each class. The parameters used are dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$. The average precision obtained is 80.56 %, with 174 out of 216 samples correctly classified.

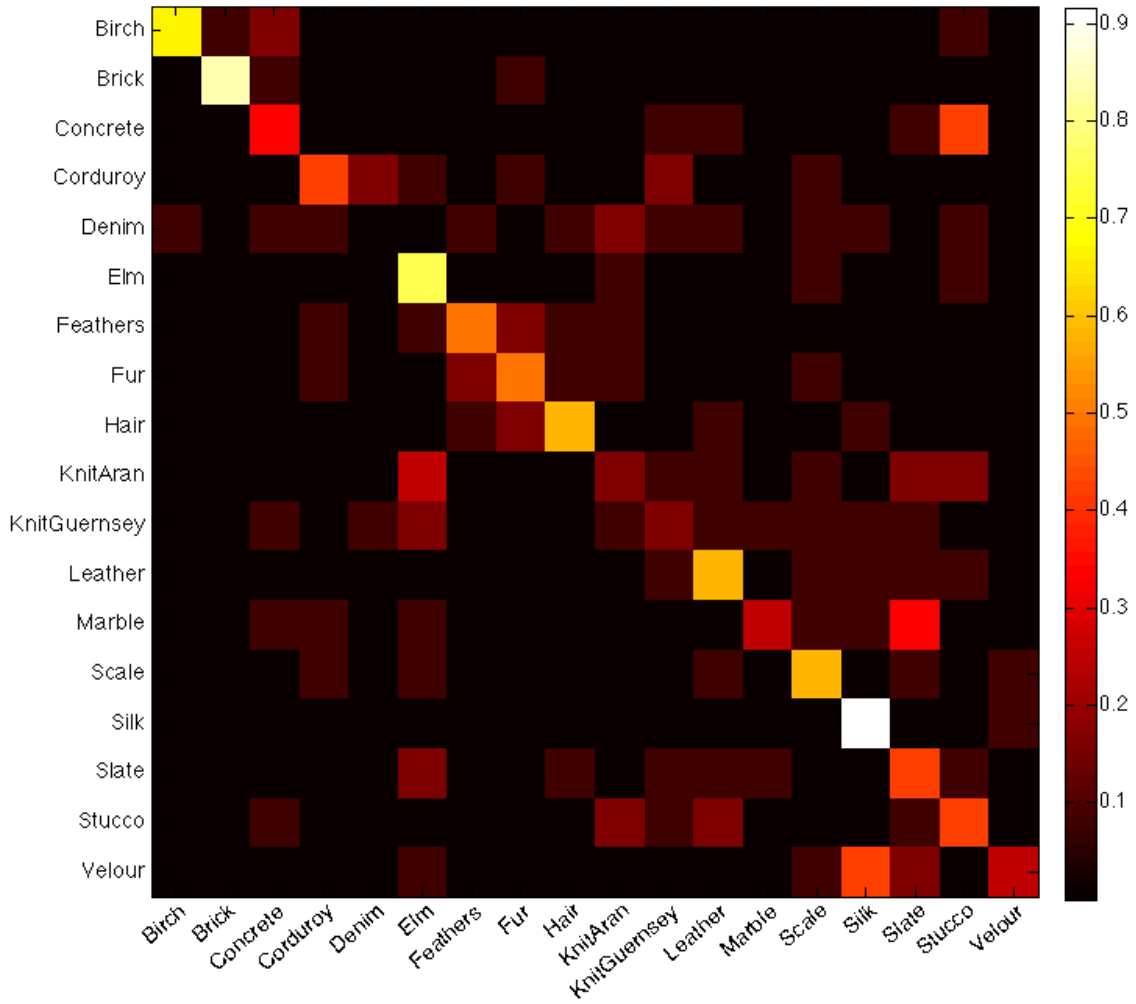


Figure 32: Heat map representing the confusion matrix of the material recognition process using *Naive Bayes* trained with real data, i.e., properties extracted from the manual markup, and tested with predicted data, i.e., properties estimated algorithmically. The results are the average of a *cross-validation* process of 12 partitions, where the test set had at least one sample of each class. The parameters used are dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$. The average precision obtained is 46.30 %, with 100 out of 216 samples correctly classified.

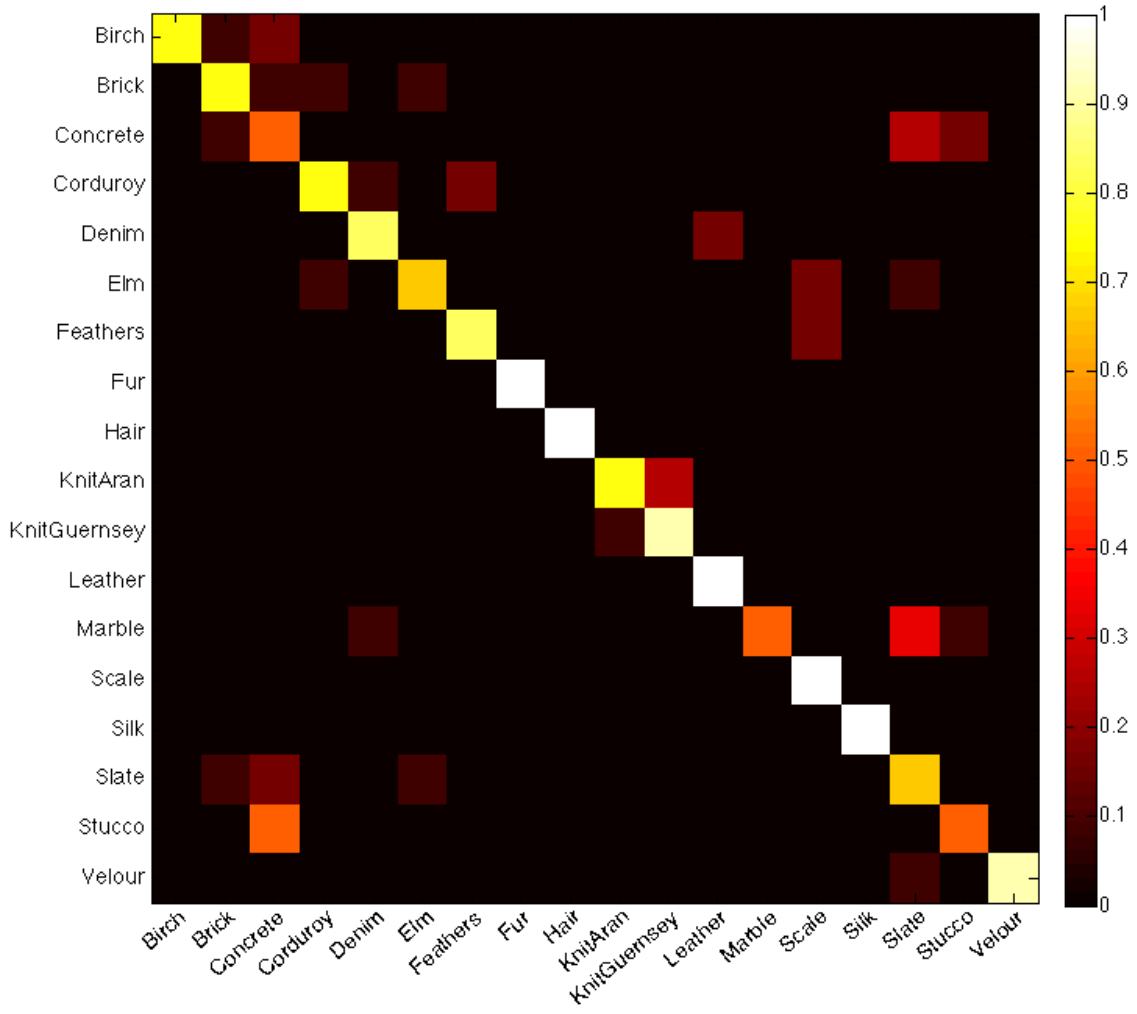


Figure 33: Heat map representing the confusion matrix of the material recognition process using *Naive Bayes* trained with real data, i.e., properties extracted from the manual markup, and tested also with predicted data. The results are the average of a *cross-validation* process of 12 partitions, where the test set had at least one sample of each class. The parameters used are dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$. The average precision obtained is 79.63 %, with 172 out of 216 samples correctly classified.

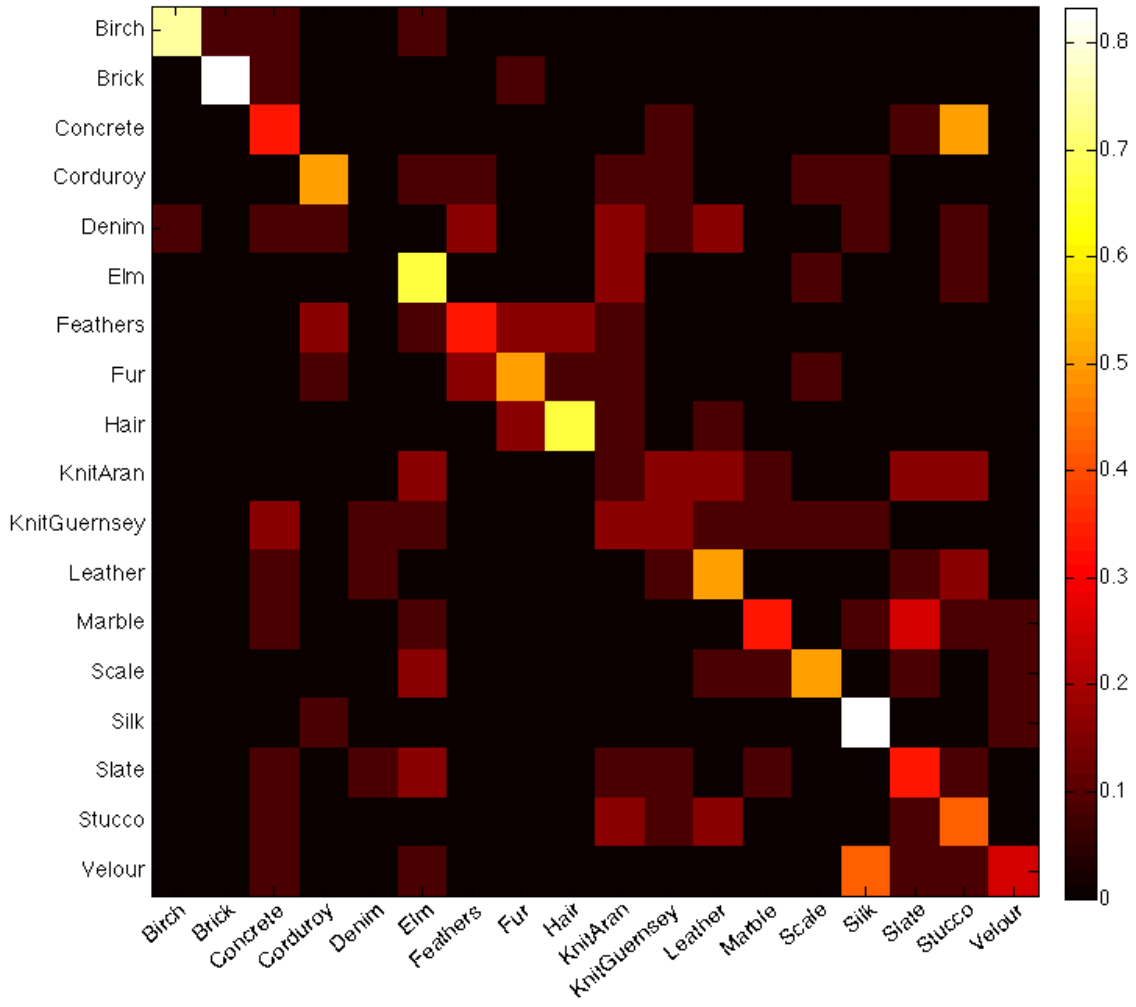


Figure 34: Heat map representing the confusion matrix of the material recognition process using *SVMs* trained with predicted data, i.e., properties estimated algorithmically, and tested also with predicted data. The results are the average of a *cross-validation* process of 12 partitions, where the test set had at least one sample of each class. The parameters used are dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$. The average precision obtained is 44.44 %, with 96 out of 216 samples correctly classified.

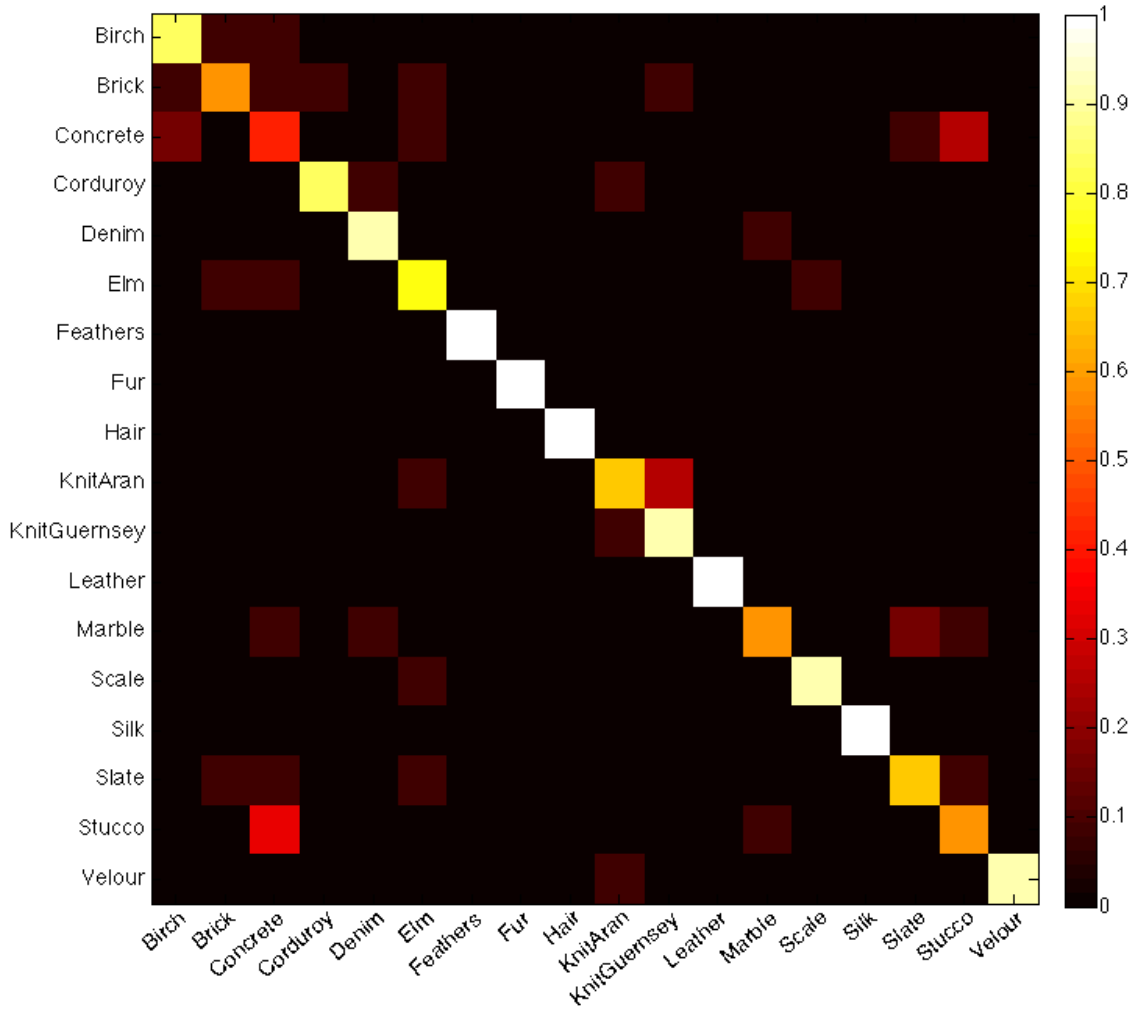


Figure 35: Heat map representing the confusion matrix of the material recognition process using *SVMs* trained with predicted data, i.e., properties estimated algorithmically, and tested with real data, i.e., properties extracted from the manual markup. The results are the average of a *cross-validation* process of 12 partitions, where the test set had at least one sample of each class. The parameters used are dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$. The average precision obtained is 81.02 %, with 175 out of 216 samples correctly classified.

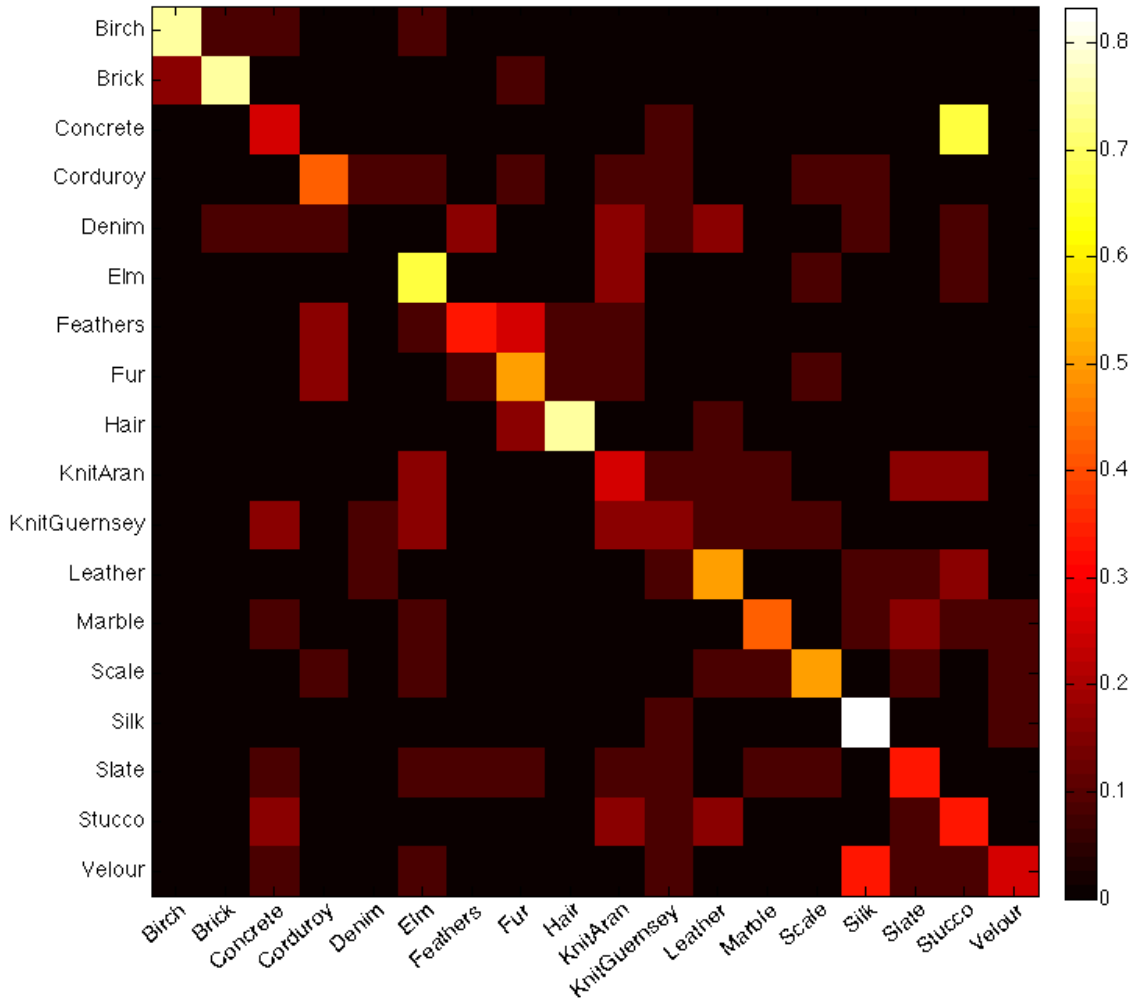


Figure 36: Heat map representing the confusion matrix of the material recognition process using *SVMs* trained with real data, i.e., properties extracted from the manual markup, and tested with predicted data, i.e., properties estimated algorithmically. The results are the average of a *cross-validation* process of 12 partitions, where the test set had at least one sample of each class. The parameters used are dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$. The average precision obtained is 44.44 %, with 96 out of 216 samples correctly classified.

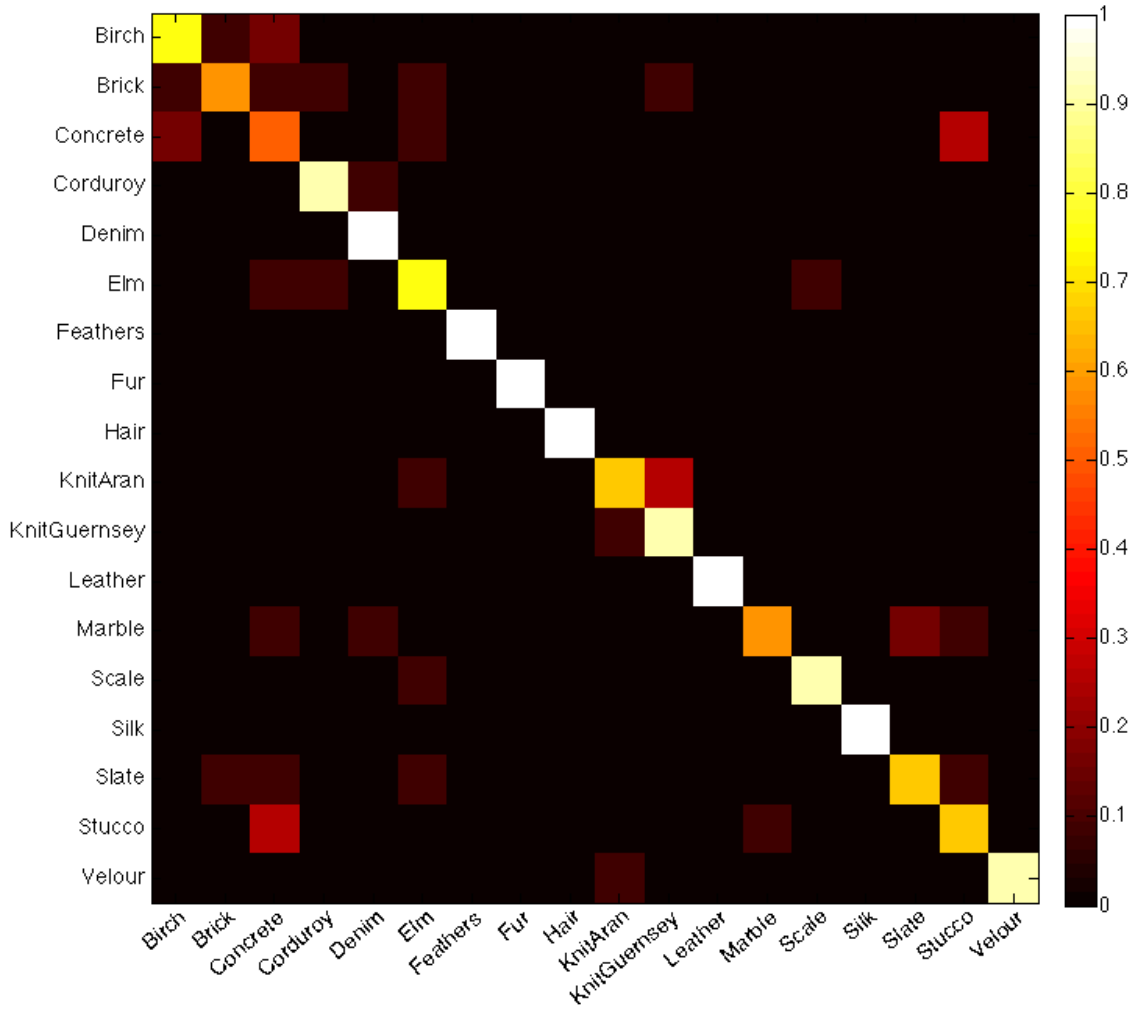


Figure 37: Heat map representing the confusion matrix of the material recognition process using *SVMs* trained with real data, i.e., properties extracted from the manual markup, and tested also with predicted data. The results are the average of a *cross-validation* process of 12 partitions, where the test set had at least one sample of each class. The parameters used are dense *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$. The average precision obtained is 82.41 %, with 178 out of 216 samples correctly classified.

6.4 Material recognition accuracy

The randomness inherent to some of the steps of techniques such as *k-means* and *SVMs* yield different results even when using the same parameters. Taking this into account, the best accuracy was obtained in an execution of the algorithm using the same set up as the graphs and bar charts shown above, i.e., *SIFT* (*PHOW*) limited to a maximum of 2000 descriptors per image, with 600 clusters and *SVM* $\lambda = 10^{-3}$. The number of correctly classifier samples were the following:

- 95 out of 216 (43.98 %) in *NB* trained with predicted data and tested with predicted data.
- 172 out of 216 (79.63 %) in *NB* trained with predicted data and tested with real data.
- 93 out of 216 (43.06 %) in *NB* trained with real data and tested with predicted data.
- 172 out of 216 (79.63 %) in *NB* trained with real data and tested with real data.
- 101 out of 216 (46.76 %) in *SVMs* trained with predicted data and tested with predicted data.
- 178 out of 216 (82.41 %) in *SVMs* trained with predicted data and tested with real data.
- 103 out of 216 (47.69 %) in *SVMs* trained with real data and tested with predicted data.
- 178 out of 216 (82.41 %) in *SVMs* trained with real data and tested with real data.

As can be seen, the trend with respect to the testing data is the same as in previous executions, and the results are in the same range, although some points above.

The result of this execution can be compared in terms of accuracy with the ones shown in [3], where the most accurate method, the one presented in the paper, gets a modest 43.5 %. Therefore, the approach introduced in this work beats all the other approaches to the problem using this dataset.

7 Conclusions and future work

Material recognition using *Computer Vision* is still an open problem in the industry. Many different approaches have been tried to solve it, being *Deep Learning* the one that, nowadays, is giving the most promising results.

Even though, it is still worth it to try different, new approaches. In this work, we have covered a solution based on classic *Machine Learning* techniques, such as *SVMs*, *k-means* and *Naive Bayes*, along with the powerful and widely used *SIFT* descriptors, but including an innovation in the use of an intermediate step based on a vocabulary of material properties.

We have analyzed the hightlights and lowlights of the approach, both from a theoretical and practical point of view, providing results using several metrics for different parts of the algorithm.

While the results look promising when using the real material properties to classify materials, further research is required to test the effects of a higher dataset and different feature descriptors methods in terms of accuracy. Even if better results are achieved in harsher conditions, the high cost of the manual markup of properties is still an important drawback of the algorithm.

Apart from the material recognition use, it cannot be denied that the information obtained in the intermediate step provides value to the classification process that could be useful even in real life applications. Besides, the theoretical possibility of one sample classification is interesting enough to motivate the continuation of this work.

References

[1] C. Cortes and V. Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[2] R. Johnson. MATLAB programming style guidelines, Oct. 2002.

[3] Z. Liao, J. Rock, Y. Wang, and D. Forsyth. Non-parametric filtering for geometric detail extraction and material representation. June 2013.

[4] S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[5] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004.

[6] A. Vedaldi and B. Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the International Conference on Multimedia*, MM '10, pages 1469–1472, New York, NY, USA, 2010. ACM.

A Markup

A.1 Birch

Birch 01:

- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, scratchy.*
- **Coarse**: *extended organized, scratchy.*

Birch 02:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *extended disorganized, scratchy.*

Birch 03:

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, scratchy.*
- **Coarse**: *extended organized, scratchy.*

Birch 04:
Watermark and big crack.

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, scratchy.*
- **Coarse**: *round, coarse.*

Birch 05:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *extended organized, scratchy.*

Considering for the medium scale the part without horizontal lines.

Birch 06:

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, scratchy.*
- **Coarse**: *extended organized, scratchy.*

Birch 07:

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, coarse.*
- **Coarse**: *extended organized, coarse.*

Birch 08:
Watermark.

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, scratchy.*
- **Coarse**: *extended organized, scratchy.*

Birch 09:

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, scratchy.*
- **Coarse**: *extended disorganized, scratchy.*

Birch 10:

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, scratchy.*
- **Coarse**: *extended organized, scratchy.*

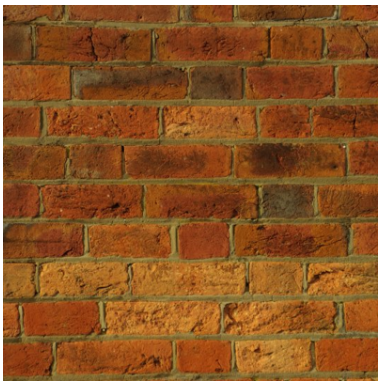
Birch 11:
Big crack.

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, scratchy.*
- **Coarse**: *round, scratchy.*

Birch 12:

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, scratchy.*
- **Coarse**: *extended organized, scratchy.*

A.2 Brick



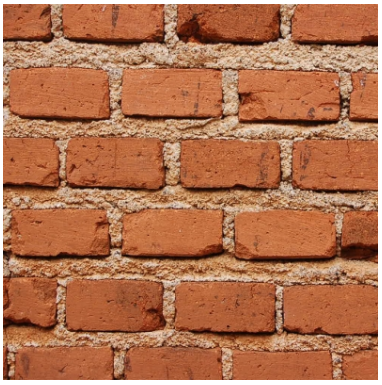
Brick 01:

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, coarse.*
- **Coarse**: *extended organized, coarse.*

Brick 02:

- **Fine**: *flat, scratchy.*
- **Medium**: *extended organized, coarse.*
- **Coarse**: *extended organized, coarse.*

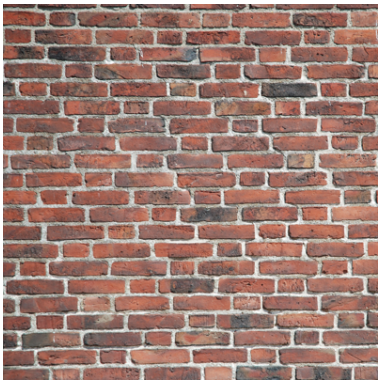
Considering as fine scale the mortar.



Brick 03:

- **Fine**: *flat, scratchy.*
- **Medium**: *flat, coarse.*
- **Coarse**: *extended organized, scratchy.*

Considering as fine scale the mortar.



Brick 04:

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, coarse.*
- **Coarse**: *extended organized, coarse.*

Brick 05:

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, coarse.*
- **Coarse**: *extended organized, coarse.*

Might also be considered bumpy at coarse and medium scale.

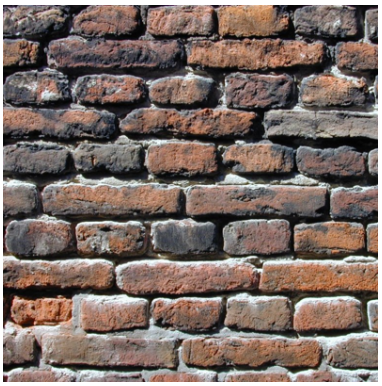
Brick 06:
Graffiti.

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended organized, smooth.*

Considering as fine scale the mortar.

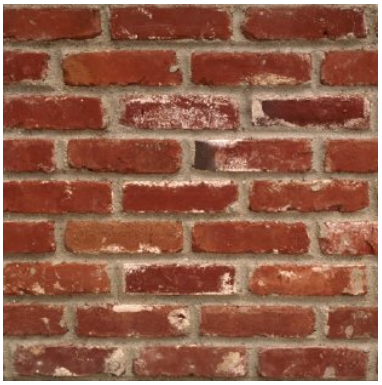
Brick 07:

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, coarse.*
- **Coarse**: *extended organized, coarse.*



Brick 08:

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *extended organized, bumpy.*



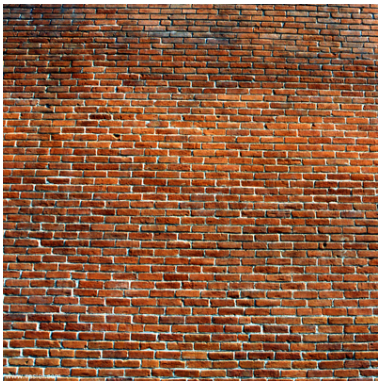
Brick 09:

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, coarse.*
- **Coarse**: *extended organized, coarse.*

Brick 10:

- **Fine**: *flat, scratchy.*
- **Medium**: *extended organized, coarse.*
- **Coarse**: *extended organized, coarse.*

Considering as fine scale the mortar.



Brick 11:

- **Fine**: *extended organized, coarse.*
- **Medium**: *extended organized, coarse.*
- **Coarse**: *extended organized, coarse.*

Brick 12:

- **Fine**: *flat, smooth.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended organized, bumpy.*

A.3 Concrete

Concrete 01:

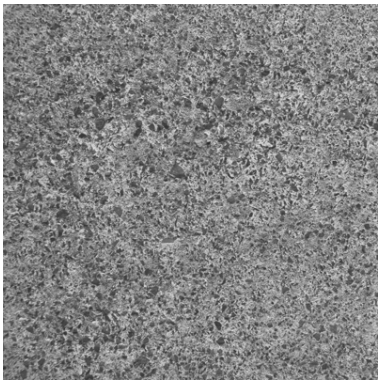
- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *flat, coarse.*

Concrete 02:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *extended disorganized, coarse.*

Concrete 03:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *extended disorganized, bumpy.*



Concrete 04:

- **Fine**: *flat, scratchy.*
- **Medium**: *flat, scratchy.*
- **Coarse**: *flat, scratchy.*

Concrete 05:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *extended organized, coarse.*

Concrete 06:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *extended organized, coarse.*

Considering that the almost straight break is organized.

Concrete 07:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *round, scratchy.*

Considering the spots of different colours to determine a round shape.

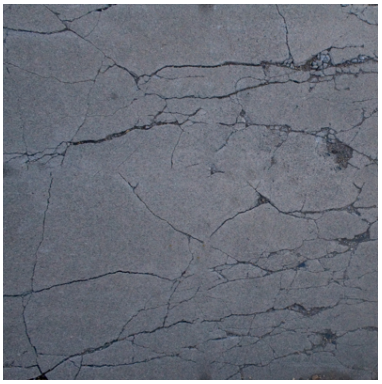
Concrete 08:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *flat, coarse.*

Considering that the patterns in the middle of the image don't affect the shape.

Concrete 09:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *flat, coarse.*



Concrete 10:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *extended disorganized, coarse.*

Concrete 11:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, scratchy.*
- **Coarse**: *flat, scratchy.*

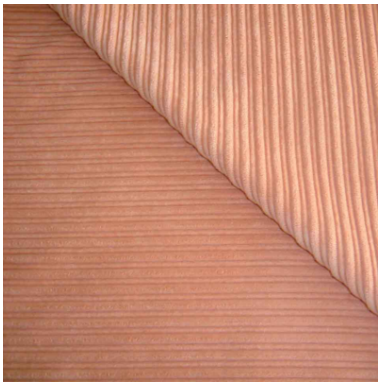
Concrete 12:

- **Fine**: *flat, scratchy.*
- **Medium**: *flat, scratchy.*
- **Coarse**: *flat, scratchy.*

A.4 Corduroy

Corduroy 01:
Logo in the right bottom corner.

- **Fine**: *extended organized, smooth.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended organized, bumpy.*



Corduroy 02:

- **Fine**: *flat, velvety.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *extended disorganized, bumpy.*

Considering that the perpendicular direction of the lines makes them disorganized.

Corduroy 03:

- **Fine**: *flat, smooth.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *extended disorganized, bumpy.*

Corduroy 04:

- **Fine**: *flat, smooth.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *extended disorganized, bumpy.*

Corduroy 05:

- **Fine**: *flat, velvety.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *extended disorganized, bumpy.*

Corduroy 06:

- **Fine**: *flat, velvety.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *round, bumpy.*

Considering the buttons as part of the course scale.

Corduroy 07:

- **Fine**: *flat, velvety.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *extended organized, bumpy.*

Corduroy 08:
Logo in the bottom left corner.

- **Fine**: *flat, velvety.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *extended organized, bumpy.*

Corduroy 09:

- **Fine**: *flat, velvety.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *extended organized, bumpy.*

Corduroy 10:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *round, bumpy.*

Considering the figures of the bottom left corner for the coarse scale.

Corduroy 11:

- **Fine**: *flat, velvety.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *extended organized, bumpy.*

Corduroy 12:

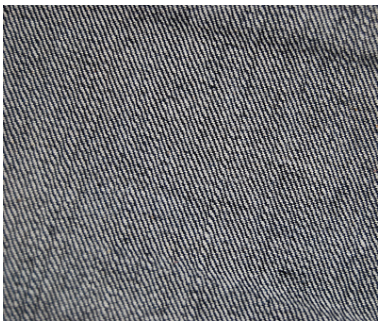
- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *round, bumpy.*

Considering the flower for the coarse scale.

A.5 Denim

Denim 01:

- **Fine**: *extended organized, smooth.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended organized, smooth.*



Denim 02:

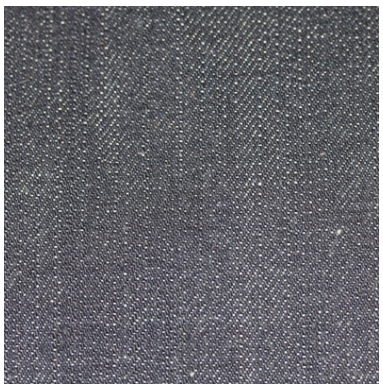
- **Fine**: *extended organized, smooth.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended organized, smooth.*

Denim 03:

- **Fine**: *extended organized, smooth.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended disorganized, bumpy.*

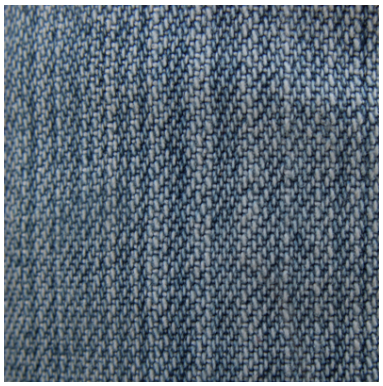
Denim 04:

- **Fine**: *extended organized, smooth.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended disorganized, bumpy.*



Denim 05:

- **Fine**: *extended organized, smooth.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended organized, smooth.*

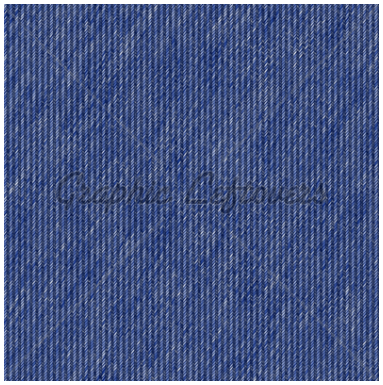


Denim 06:

- **Fine**: *round, smooth.*
- **Medium**: *round, smooth.*
- **Coarse**: *round, smooth.*

Denim 07:

- **Fine**: *extended organized, smooth.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended disorganized, smooth.*



Denim 08:
Watermark.

- **Fine**: *extended organized, smooth.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended organized, smooth.*

Denim 09:

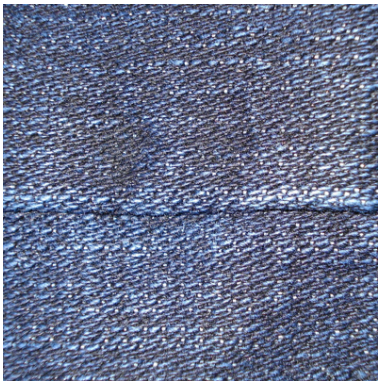
- **Fine**: *round, smooth.*
- **Medium**: *round, smooth.*
- **Coarse**: *round, smooth.*

Denim 10:

- **Fine**: *extended organized, smooth.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended disorganized, bumpy.*

Denim 11:

- **Fine**: *round, smooth.*
- **Medium**: *round, smooth.*
- **Coarse**: *round, bumpy.*



Denim 12:

- **Fine**: *round, smooth.*
- **Medium**: *round, smooth.*
- **Coarse**: *round, bumpy.*

A.6 Elm

Elm 01:

- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, bumpy.*
- **Coarse**: *extended disorganized, bumpy.*

Considering the bark to be disorganized.

Elm 02:

- **Fine**: *flat, coarse.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

Elm 03:

- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, bumpy.*
- **Coarse**: *extended disorganized, bumpy.*

Elm 04:

- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, bumpy.*
- **Coarse**: *extended disorganized, bumpy.*

Elm 05:

- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, bumpy.*
- **Coarse**: *extended disorganized, bumpy.*

Elm 06:

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *extended organized, bumpy.*

Considering the bark as organized.

Elm 07:

- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, bumpy.*
- **Coarse**: *extended disorganized, bumpy.*

Elm 08:

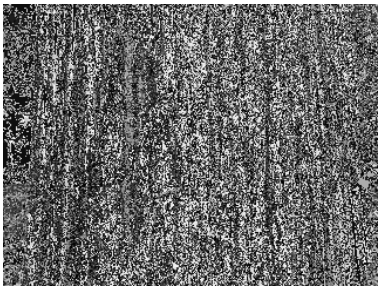
- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, bumpy.*
- **Coarse**: *extended disorganized, bumpy.*

Elm 09:

- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, bumpy.*
- **Coarse**: *extended disorganized, bumpy.*

Elm 10:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *extended disorganized, bumpy.*



Elm 11:
Extremely low quality image.

- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, bumpy.*
- **Coarse**: *extended disorganized, bumpy.*

Elm 12:

- **Fine**: *flat, coarse.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

A.7 Feathers

Feathers 01:
Watermark.

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *round, feathery.*

Feathers 02:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *round, feathery.*

Feathers 03:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *round, feathery.*

Feathers 04:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *extended disorganized, feathery.*

Feathers 05:
Watermark.

- **Fine**: *extended organized, bumpy.*
- **Medium**: *round, feathery.*
- **Coarse**: *round, feathery.*

Feathers 06:

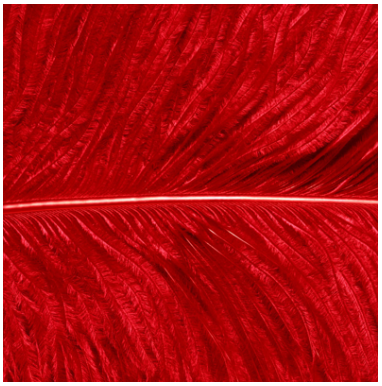
- **Fine**: *flat, smooth.*
- **Medium**: *round, feathery.*
- **Coarse**: *round, feathery.*

Feathers 07:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended disorganized, feathery.*
- **Coarse**: *extended disorganized, feathery.*

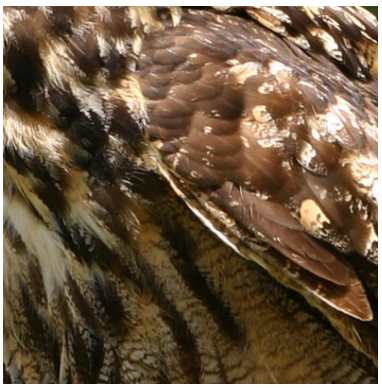
Feathers 08:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *extended disorganized, feathery.*



Feathers 09:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, bumpy.*
- **Coarse**: *extended disorganized, feathery.*



Feathers 10:

- **Fine**: *flat, smooth.*
- **Medium**: *round, feathery.*
- **Coarse**: *round, feathery.*

Considering the wing for the medium scale.

Feathers 11:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, feathery.*
- **Coarse**: *extended organized, feathery.*

Feathers 12:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *round, feathery.*
- **Coarse**: *round, feathery.*

A.8 Fur

Considering that the pattern that follows the fur is, at least, at low scale, organized.

Fur 01:
Watermark.

- **Fine**: *extended organized, furry.*
- **Medium**: *extended organized, furry.*
- **Coarse**: *extended disorganized, furry.*

Fur 02:

- **Fine**: *extended organized, furry.*
- **Medium**: *extended organized, furry.*
- **Coarse**: *extended disorganized, furry.*

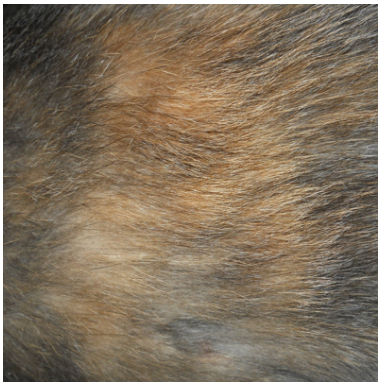
Fur 03:

- **Fine**: *extended organized, furry.*
- **Medium**: *extended organized, furry.*
- **Coarse**: *round, furry.*

Considering the spots as elements of round shape.

Fur 04:

- **Fine**: *extended organized, furry.*
- **Medium**: *extended organized, furry.*
- **Coarse**: *extended organized, furry.*



Fur 05:

- **Fine**: *extended organized, furry.*
- **Medium**: *extended organized, furry.*
- **Coarse**: *extended disorganized, furry.*

Fur 06:

- **Fine**: *extended organized, furry.*
- **Medium**: *extended organized, furry.*
- **Coarse**: *extended disorganized, furry.*

Fur 07:

- **Fine**: *extended organized, furry.*
- **Medium**: *extended disorganized, furry.*
- **Coarse**: *extended disorganized, furry.*

Fur 08:

- **Fine**: *extended organized, furry.*
- **Medium**: *extended organized, furry.*
- **Coarse**: *extended disorganized, furry.*

Fur 09:

- **Fine**: *extended organized, furry.*
- **Medium**: *extended disorganized, furry.*
- **Coarse**: *extended disorganized, furry.*

Fur 10:

- **Fine**: *extended organized, furry.*
- **Medium**: *extended organized, furry.*
- **Coarse**: *extended disorganized, furry.*

Fur 11:
Same as Fur 03 with different color.

- **Fine**: *extended organized, furry.*
- **Medium**: *extended organized, furry.*
- **Coarse**: *round, furry.*

Fur 12:

- **Fine**: *extended organized, furry.*
- **Medium**: *extended organized, furry.*
- **Coarse**: *extended disorganized, furry.*

A.9 Hair

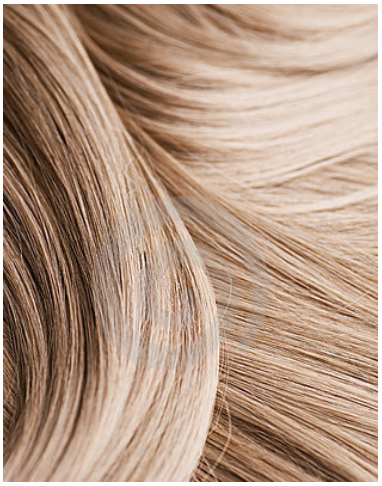
Considering that the touch of hair is smooth at a high scale.

Hair 01:
Watermark.

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended disorganized, smooth.*

Hair 02:
Watermark.

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended disorganized, smooth.*
- **Coarse**: *extended disorganized, smooth.*



Hair 03:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended disorganized, smooth.*

Hair 04:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended disorganized, smooth.*

Hair 05:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended organized, smooth.*

Hair 06:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended organized, smooth.*

Hair 07:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended disorganized, smooth.*

Hair 08:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended disorganized, smooth.*
- **Coarse**: *extended disorganized, smooth.*

Hair 09:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended disorganized, smooth.*

Hair 10:
Watermark.

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended disorganized, smooth.*

Hair 11:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended organized, smooth.*

Hair 12:

- **Fine**: *extended organized, bumpy.*
- **Medium**: *extended organized, smooth.*
- **Coarse**: *extended disorganized, smooth.*

A.10 KnitAran

Considering that non-linear patterns are disorganized.

KnitAran 01:

- **Fine**: *extended organized, coarse.*
- **Medium**: *extended disorganized, bumpy.*
- **Coarse**: *extended disorganized, bumpy.*

KnitAran 02:

- **Fine**: *extended disorganized, coarse.*
- **Medium**: *extended disorganized, bumpy.*
- **Coarse**: *extended disorganized, bumpy.*

KnitAran 03:

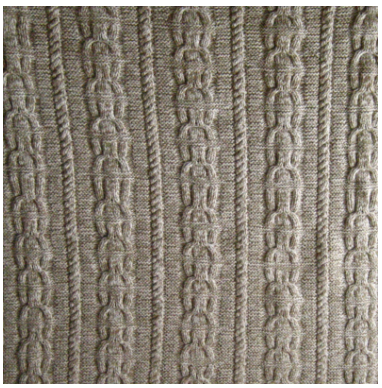
- **Fine**: *extended organized, coarse.*
- **Medium**: *extended disorganized, bumpy.*
- **Coarse**: *extended disorganized, bumpy.*

Considering that the round parts of the pattern are extended disorganized.

KnitAran 04:

- **Fine**: *extended organized, coarse.*
- **Medium**: *extended disorganized, bumpy.*
- **Coarse**: *round, bumpy.*

Considering the buttons at the coarse level.



KnitAran 05:

- **Fine**: *extended organized, coarse.*
- **Medium**: *extended disorganized, bumpy.*
- **Coarse**: *round, bumpy.*

KnitAran 06:

- **Fine**: *extended organized, coarse.*
- **Medium**: *extended disorganized, bumpy.*
- **Coarse**: *round, bumpy.*

KnitAran 07:

- **Fine**: *extended organized, coarse.*
- **Medium**: *extended organized, coarse.*
- **Coarse**: *extended disorganized, bumpy.*

KnitAran 08:

- **Fine**: *extended disorganized, coarse.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *extended disorganized, bumpy.*

KnitAran 09:

- **Fine**: *extended organized, coarse.*
- **Medium**: *extended disorganized, bumpy.*
- **Coarse**: *extended disorganized, bumpy.*

KnitAran 10:

- **Fine**: *extended organized, coarse.*
- **Medium**: *extended organized, coarse.*
- **Coarse**: *extended disorganized, bumpy.*

KnitAran 11:

- **Fine**: *extended organized, coarse.*
- **Medium**: *extended disorganized, bumpy.*
- **Coarse**: *round, bumpy.*

Considering the buttons at the coarse level.

KnitAran 12:

- **Fine**: *extended organized, coarse.*
- **Medium**: *extended disorganized, bumpy.*
- **Coarse**: *extended disorganized, bumpy.*

A.11 KnitGuernsey

Considering that non-linear patterns are disorganized.

KnitGuernsey 01:

- **Fine**: *extended organized, coarse.*
- **Medium**: *extended organized, coarse.*
- **Coarse**: *extended disorganized, bumpy.*

KnitGuernsey 02:

- **Fine**: *extended organized, coarse.*
- **Medium**: *extended organized, coarse.*
- **Coarse**: *extended disorganized, coarse.*

Ignoring the table behind.

KnitGuernsey 03:

- **Fine**: *round, coarse.*
- **Medium**: *round, coarse.*
- **Coarse**: *extended disorganized, bumpy.*

KnitGuernsey 04:

- **Fine**: *round, coarse.*
- **Medium**: *round, coarse.*
- **Coarse**: *extended organized, bumpy.*

KnitGuernsey 05:

- **Fine**: *extended organized, coarse.*
- **Medium**: *extended organized, coarse.*
- **Coarse**: *extended disorganized, bumpy.*

KnitGuernsey 06:

- **Fine**: *extended disorganized, coarse.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *extended disorganized, bumpy.*

KnitGuernsey 07:

- **Fine**: *round, coarse.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *extended disorganized, bumpy.*

KnitGuernsey 08:

- **Fine**: *extended organized, coarse.*
- **Medium**: *extended organized, coarse.*
- **Coarse**: *extended disorganized, bumpy.*

KnitGuernsey 09:

- **Fine**: *round, coarse.*
- **Medium**: *round, coarse.*
- **Coarse**: *extended disorganized, bumpy.*

KnitGuernsey 10:

- **Fine**: *round, coarse.*
- **Medium**: *round, coarse.*
- **Coarse**: *extended disorganized, bumpy.*

KnitGuernsey 11:

- **Fine**: *round, coarse.*
- **Medium**: *round, coarse.*
- **Coarse**: *extended disorganized, bumpy.*

KnitGuernsey 12:

- **Fine**: *round, coarse.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *extended disorganized, bumpy.*

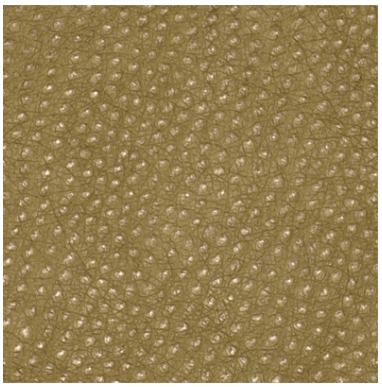
A.12 Leather

Leather 01:

- **Fine**: *round, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

Leather 02:

- **Fine**: *round, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*



Leather 03:

- **Fine**: *round, coarse.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

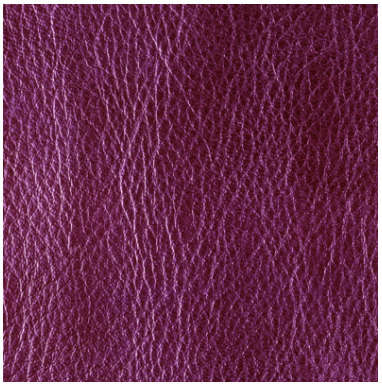
Leather 04:

- **Fine**: *round, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

Leather 05:

- **Fine**: *round, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

Same as leather 04.



Leather 06:

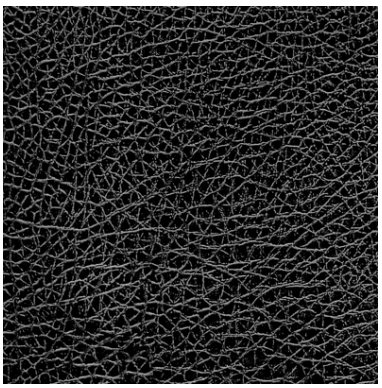
- **Fine**: *round, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

Leather 07:

- **Fine**: *round, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

Leather 08:

- **Fine**: *round, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

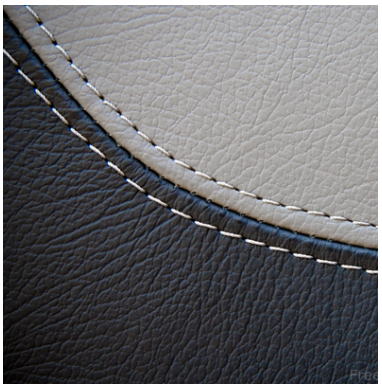


Leather 09:

- **Fine**: *round, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

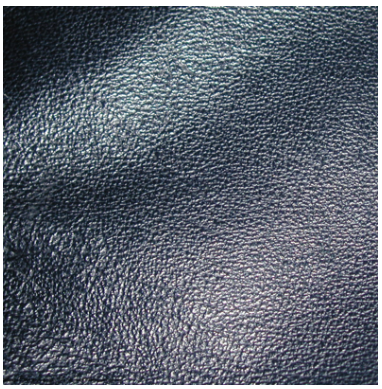
Leather 10:

- **Fine**: *round, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*



Leather 11:

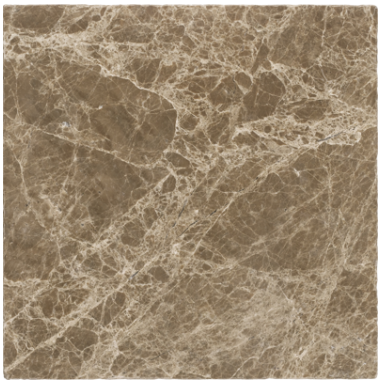
- **Fine**: *round, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *extended organized, bumpy.*



Leather 12:

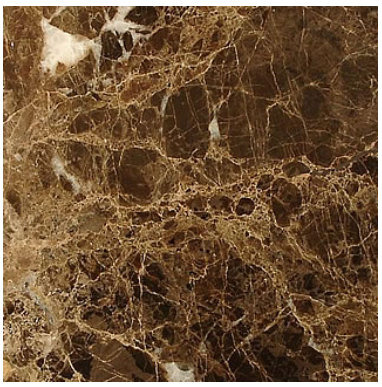
- **Fine**: *round, coarse.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

A.13 Marble



Marble 01:

- **Fine**: *flat, smooth.*
- **Medium**: *extended disorganized, smooth.*
- **Coarse**: *extended disorganized, smooth.*



Marble 02:

- **Fine**: *flat, smooth.*
- **Medium**: *extended disorganized, smooth.*
- **Coarse**: *extended disorganized, smooth.*

Marble 03:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *extended disorganized, coarse.*

Marble 04:

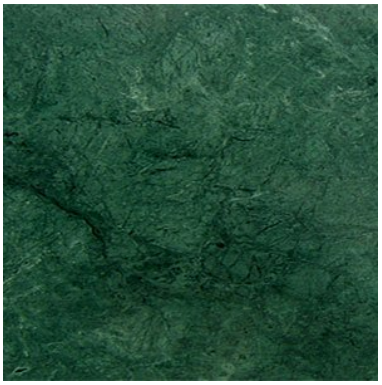
- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *extended disorganized, coarse.*

Marble 05:

- **Fine**: *flat, smooth.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *extended disorganized, coarse.*

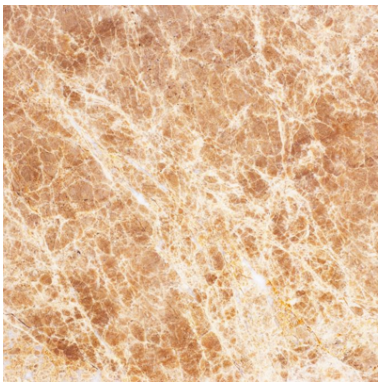
Marble 06:

- **Fine**: *extended disorganized, coarse.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *extended disorganized, coarse.*



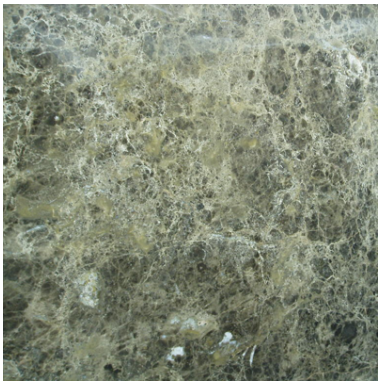
Marble 07:

- **Fine**: *extended disorganized, coarse.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *extended disorganized, coarse.*



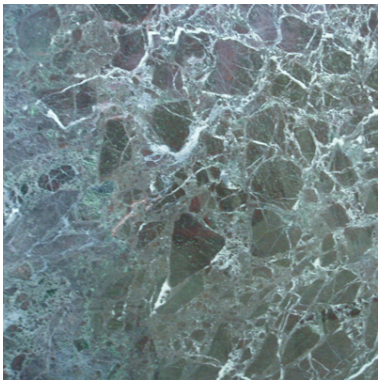
Marble 08:

- **Fine**: *extended disorganized, smooth.*
- **Medium**: *extended disorganized, smooth.*
- **Coarse**: *extended disorganized, smooth.*



Marble 09:

- **Fine**: *round, smooth.*
- **Medium**: *round, smooth.*
- **Coarse**: *round, smooth.*

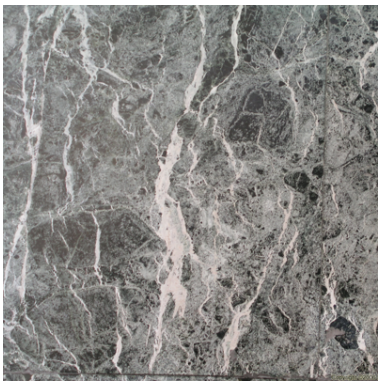


Marble 10:

- **Fine**: *flat, smooth.*
- **Medium**: *extended disorganized, smooth.*
- **Coarse**: *round, smooth.*

Marble 11:

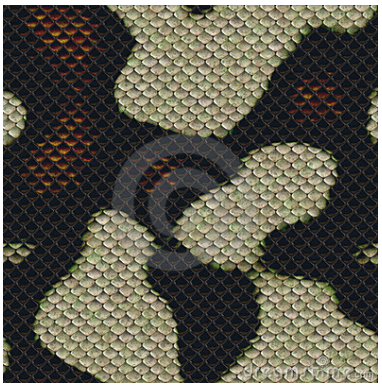
- **Fine**: *round, coarse.*
- **Medium**: *round, coarse.*
- **Coarse**: *round, coarse.*



Marble 12:

- **Fine**: *flat, smooth.*
- **Medium**: *extended disorganized, smooth.*
- **Coarse**: *extended disorganized, smooth.*

A.14 Scale



Scale 01:

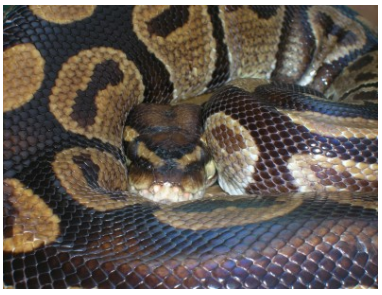
- **Fine**: *flat, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

Scale 02:

- **Fine**: *flat, coarse.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

Scale 03:

- **Fine**: *flat, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*



Scale 04:

- **Fine**: *flat, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

Scale 05:

- **Fine**: *flat, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

Scale 06:
Watermark.

- **Fine**: *flat, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

Scale 07:
Watermark.

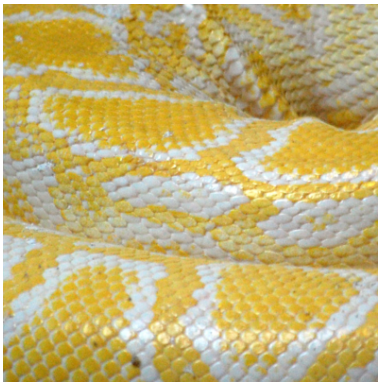
- **Fine**: *flat, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

Scale 08:

- **Fine**: *flat, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

Scale 09:

- **Fine**: *flat, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*



Scale 10:

- **Fine**: *flat, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

Scale 11:

- **Fine**: *flat, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

Scale 12:

- **Fine**: *flat, smooth.*
- **Medium**: *round, bumpy.*
- **Coarse**: *round, bumpy.*

A.15 Silk

Silk 01:

- **Fine**: *flat, smooth.*
- **Medium**: *flat, smooth.*
- **Coarse**: *extended disorganized, bumpy.*

Silk 02:

- **Fine**: *flat, smooth.*
- **Medium**: *flat, smooth.*
- **Coarse**: *extended disorganized, bumpy.*

Silk 03:

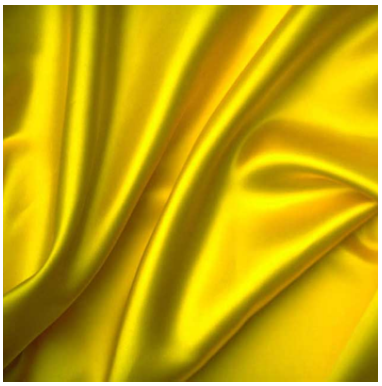
- **Fine**: *flat, smooth.*
- **Medium**: *flat, smooth.*
- **Coarse**: *extended disorganized, bumpy.*

Silk 04:

- **Fine**: *extended organized, smooth.*
- **Medium**: *flat, smooth.*
- **Coarse**: *extended disorganized, bumpy.*

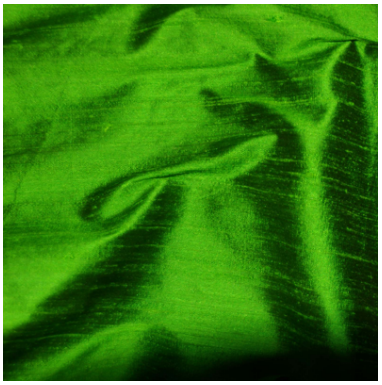
Silk 05:

- **Fine**: *flat, smooth.*
- **Medium**: *flat, smooth.*
- **Coarse**: *extended disorganized, bumpy.*



Silk 06:

- **Fine**: *flat, smooth.*
- **Medium**: *flat, smooth.*
- **Coarse**: *extended disorganized, bumpy.*

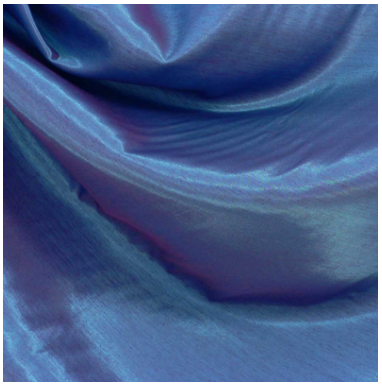


Silk 07:

- **Fine**: *flat, smooth.*
- **Medium**: *flat, smooth.*
- **Coarse**: *extended disorganized, bumpy.*

Silk 08:

- **Fine**: *flat, smooth.*
- **Medium**: *flat, smooth.*
- **Coarse**: *extended disorganized, bumpy.*



Silk 09:

- **Fine**: *flat, smooth.*
- **Medium**: *flat, smooth.*
- **Coarse**: *extended disorganized, bumpy.*

Silk 10:

- **Fine**: *extended organized, smooth.*
- **Medium**: *flat, smooth.*
- **Coarse**: *extended disorganized, bumpy.*

Silk 11:

- **Fine**: *flat, smooth.*
- **Medium**: *flat, smooth.*
- **Coarse**: *extended disorganized, bumpy.*

Silk 12:

- **Fine**: *flat, smooth.*
- **Medium**: *flat, smooth.*
- **Coarse**: *extended disorganized, bumpy.*

A.16 Slate

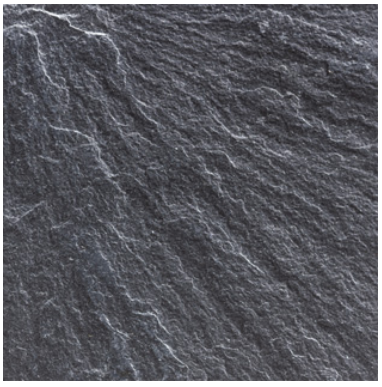
Slate 01:

- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *extended disorganized, coarse.*

Slate 02:

?

- **Fine**: *flat, coarse.*
- **Medium**: *extended organized, coarse.*
- **Coarse**: *extended organized, coarse.*



Slate 03:

- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *extended disorganized, coarse.*

Slate 04:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *extended disorganized, coarse.*

Slate 05:

- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *extended disorganized, coarse.*

Slate 06:

- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *extended disorganized, coarse.*

Slate 07:

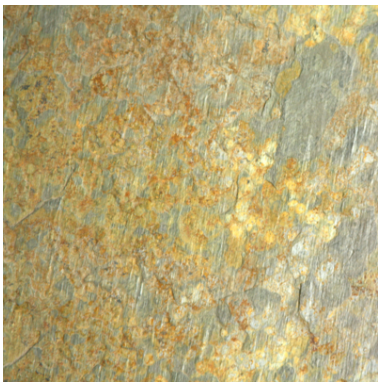
- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *extended disorganized, bumpy.*

Slate 08:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *flat, coarse.*

Slate 09:

- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *extended disorganized, coarse.*



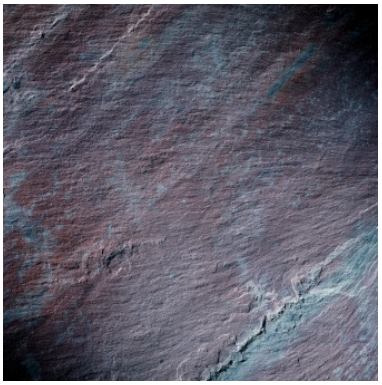
Slate 10:

- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *extended disorganized, coarse.*

Slate 11:
Watermark.

- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *extended disorganized, coarse.*

Same as slate 09.



Slate 12:

- **Fine**: *flat, coarse.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *extended disorganized, coarse.*

A.17 Stucco

Stucco 01:

- **Fine**: *flat, smooth.*
- **Medium**: *round, coarse.*
- **Coarse**: *round, coarse.*

Stucco 02:
Watermark.

- **Fine**: *round, coarse.*
- **Medium**: *round, coarse.*
- **Coarse**: *round, coarse.*

Stucco 03:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *flat, coarse.*

Stucco 04:

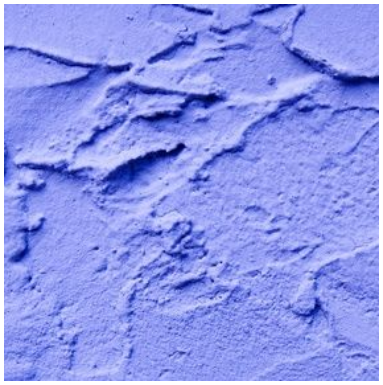
- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *flat, coarse.*

Stucco 05:

- **Fine**: *flat, coarse.*
- **Medium**: *round, coarse.*
- **Coarse**: *round, coarse.*

Stucco 06:

- **Fine**: *flat, coarse.*
- **Medium**: *round, coarse.*
- **Coarse**: *round, coarse.*



Stucco 07:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *extended disorganized, bumpy.*

Stucco 08:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *flat, coarse.*

Stucco 09:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *flat, coarse.*

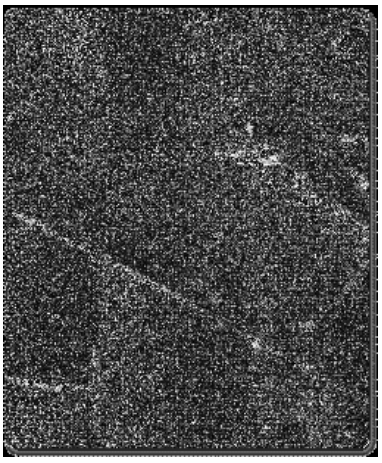
Stucco 10:

- **Fine**: *flat, coarse.*
- **Medium**: *round, coarse.*
- **Coarse**: *round, coarse.*

Same as stucco 05.

Stucco 11:

- **Fine**: *flat, coarse.*
- **Medium**: *round, coarse.*
- **Coarse**: *round, coarse.*



Stucco 12:

- **Fine**: *flat, coarse.*
- **Medium**: *flat, coarse.*
- **Coarse**: *flat, coarse.*

A.18 Velour

Velour 01:

- **Fine**: *flat, velvety.*
- **Medium**: *flat, velvety.*
- **Coarse**: *flat, velvety.*

Velour 02:

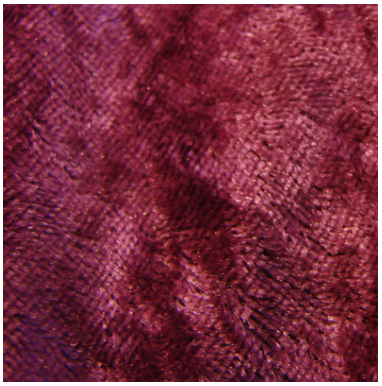
- **Fine**: *flat, velvety.*
- **Medium**: *flat, velvety.*
- **Coarse**: *extended disorganized, bumpy.*

Velour 03:

- **Fine**: *flat, velvety.*
- **Medium**: *flat, velvety.*
- **Coarse**: *extended disorganized, bumpy.*

Velour 04:

- **Fine**: *flat, velvety.*
- **Medium**: *flat, velvety.*
- **Coarse**: *extended disorganized, bumpy.*



Velour 05:

- **Fine**: *extended disorganized, coarse.*
- **Medium**: *extended disorganized, coarse.*
- **Coarse**: *flat, velvety.*

Velour 06:

- **Fine**: *flat, velvety.*
- **Medium**: *flat, velvety.*
- **Coarse**: *extended organized, bumpy.*

Velour 07:

- **Fine**: *flat, velvety.*
- **Medium**: *flat, velvety.*
- **Coarse**: *extended disorganized, bumpy.*

Velour 08:

- **Fine**: *flat, velvety.*
- **Medium**: *flat, velvety.*
- **Coarse**: *extended disorganized, bumpy.*

Velour 09:

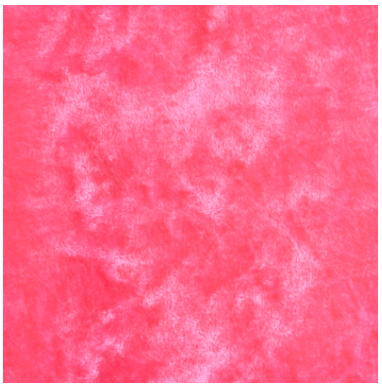
- **Fine**: *flat, velvety.*
- **Medium**: *flat, velvety.*
- **Coarse**: *extended disorganized, bumpy.*

Velour 10:

- **Fine**: *flat, velvety.*
- **Medium**: *flat, velvety.*
- **Coarse**: *extended disorganized, bumpy.*

Velour 11:

- **Fine**: *flat, velvety.*
- **Medium**: *flat, velvety.*
- **Coarse**: *extended organized, bumpy.*



Velour 12:

- **Fine**: *flat, velvety.*
- **Medium**: *flat, velvety.*
- **Coarse**: *flat, velvety.*

B Materials by property

B.1 Coarse scale - Bumpy touch images

Figure 38: **Coarse scale - Bumpy touch images**

B.2 Coarse scale - Coarse touch images

Figure 39: **Coarse scale - Coarse touch images**

B.3 Coarse scale - Extended disorganized shape images

Figure 40: **Coarse scale - Extended disorganized shape images**

B.4 Coarse scale - Extended organized shape images

Figure 41: **Coarse scale - Extended organized shape images**

B.5 Coarse scale - Feathery touch images

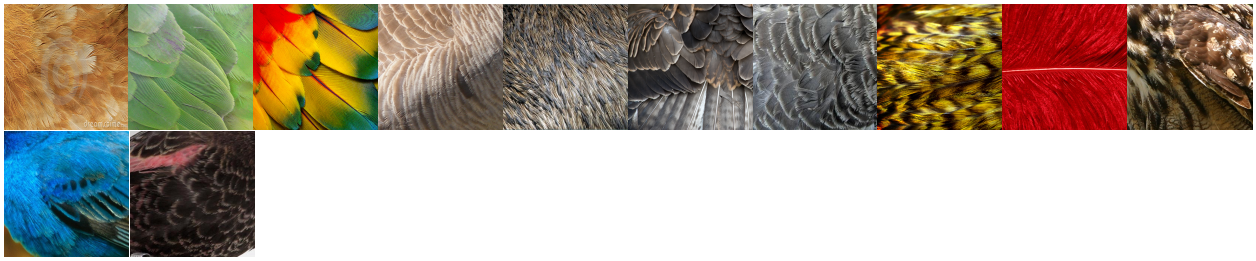


Figure 42: **Coarse scale - Feathery touch images**

B.6 Coarse scale - Flat shape images

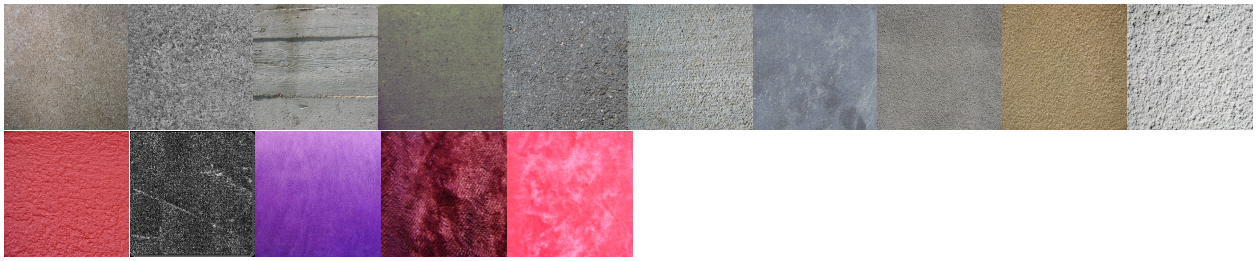


Figure 43: **Coarse scale - Flat shape images**

B.7 Coarse scale - Furry touch images

Figure 44: **Coarse scale - Furry touch images**

B.8 Coarse scale - Round shape images

Figure 45: **Coarse scale - Round shape images**

B.9 Coarse scale - Scratchy touch images

Figure 46: **Coarse scale - Scratchy touch images**

B.10 Coarse scale - Smooth touch images

Figure 47: **Coarse scale - Smooth touch images**

B.11 Coarse scale - Velvety touch images

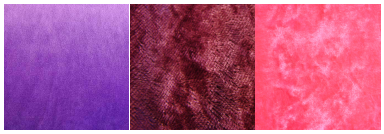


Figure 48: **Coarse scale - Velvety touch images**

B.12 Fine scale - Bumpy touch images

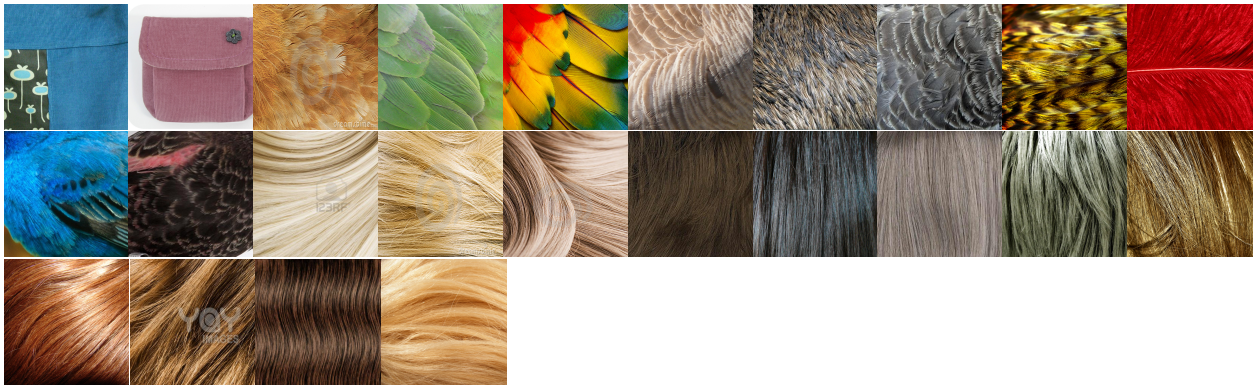


Figure 49: **Fine scale - Bumpy touch images**

B.13 Fine scale - Coarse touch images

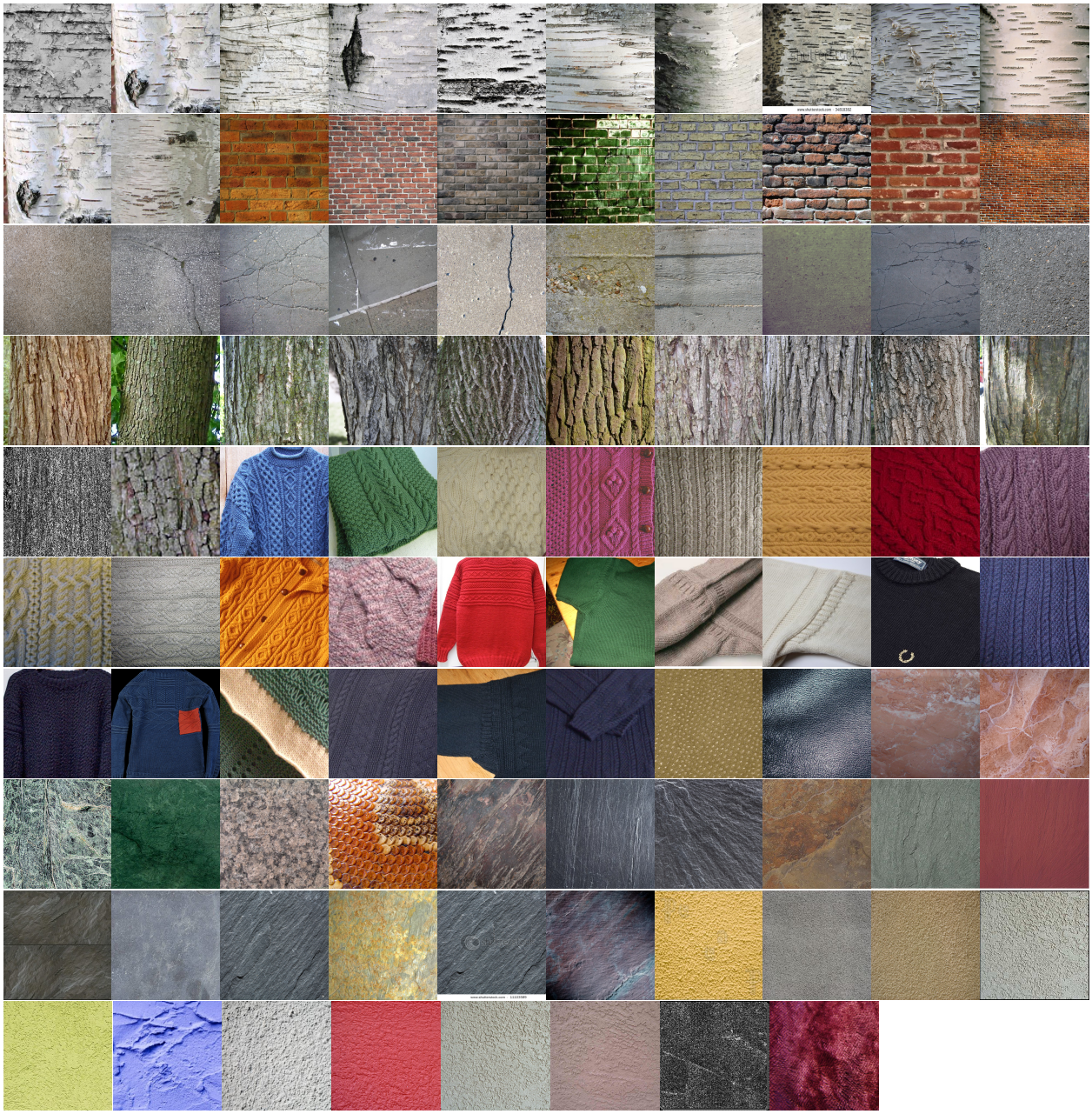


Figure 50: **Fine scale - Coarse touch images**

B.14 Fine scale - Extended disorganized shape images

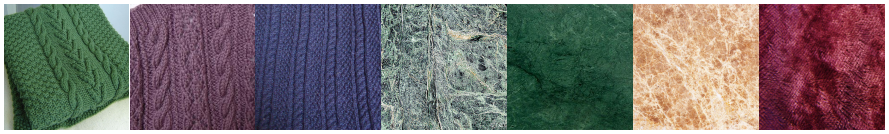


Figure 51: **Fine scale - Extended disorganized shape images**

B.15 Fine scale - Extended organized shape images

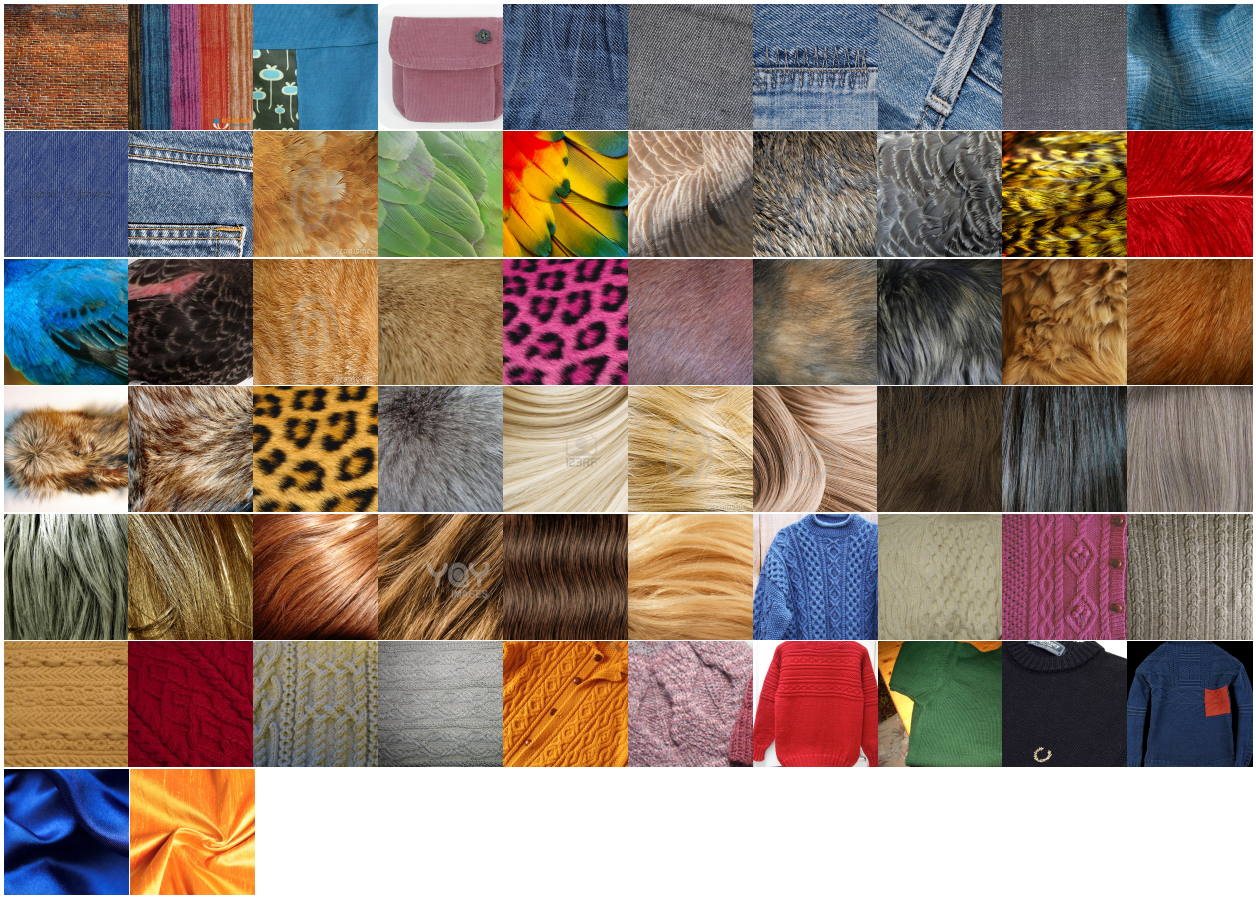


Figure 52: **Fine scale - Extended organized shape images**

B.16 Fine scale - Flat shape images

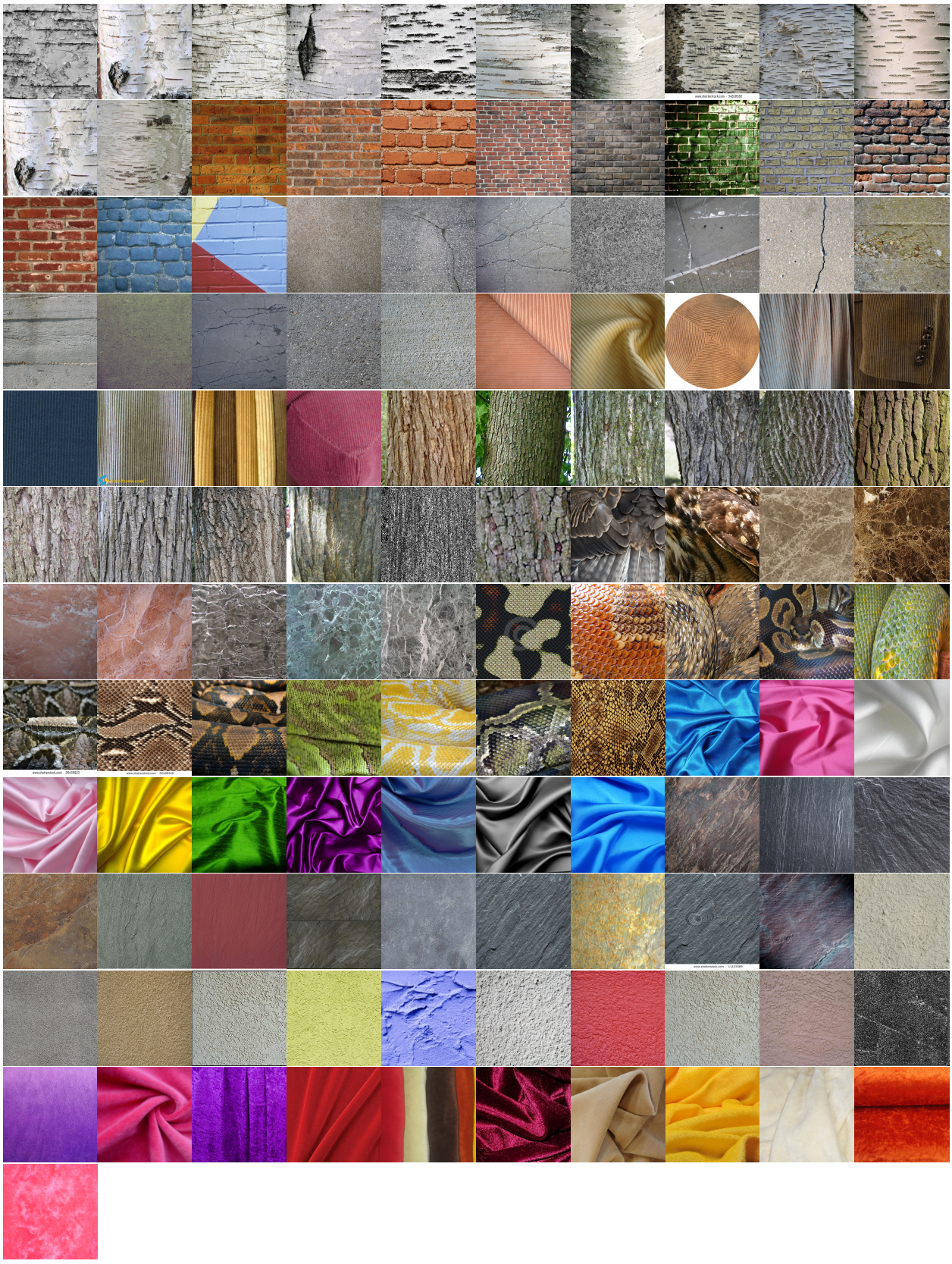


Figure 53: **Fine scale - Flat shape images**

B.17 Fine scale - Furry touch images

Figure 54: **Fine scale - Furry touch images**

B.18 Fine scale - Round shape images

Figure 55: **Fine scale - Round shape images**

B.19 Fine scale - Scratchy touch images

Figure 56: **Fine scale - Scratchy touch images**

B.20 Fine scale - Smooth touch images

Figure 57: **Fine scale - Smooth touch images**

B.21 Fine scale - Velvety touch images

Figure 58: **Fine scale - Velvety touch images**

B.22 Medium scale - Bumpy touch images

Figure 59: **Medium scale - Bumpy touch images**

B.23 Medium scale - Coarse touch images

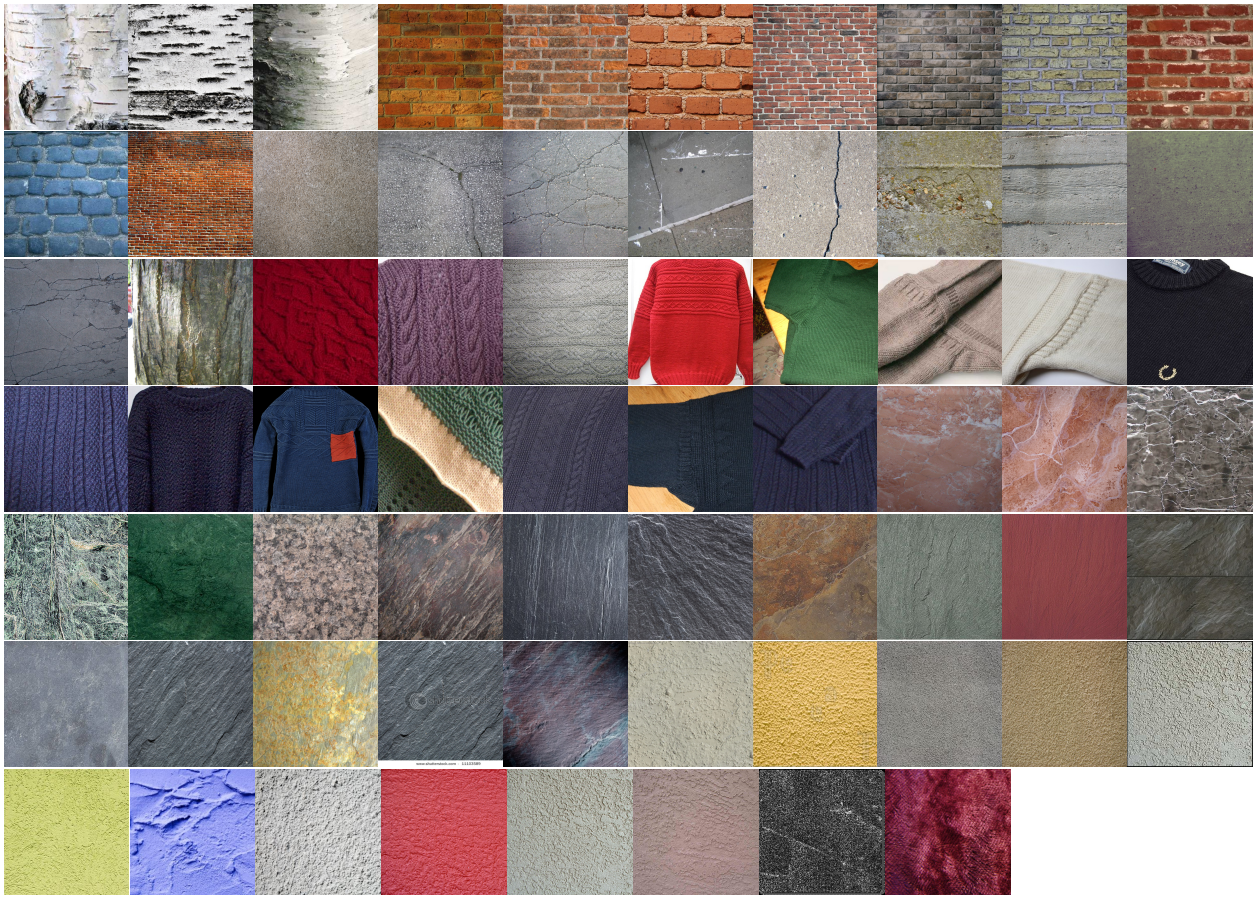


Figure 60: **Medium scale - Coarse touch images**

B.24 Medium scale - Extended disorganized shape images

Figure 61: **Medium scale - Extended disorganized shape images**

B.25 Medium scale - Extended organized shape images

Figure 62: **Medium scale - Extended organized shape images**

B.26 Medium scale - Feathery touch images

Figure 63: **Medium scale - Feathery touch images**

B.27 Medium scale - Flat shape images

Figure 64: **Medium scale - Flat shape images**

B.28 Medium scale - Furry touch images

Figure 65: **Medium scale - Furry touch images**

B.29 Medium scale - Round shape images

Figure 66: **Medium scale - Round shape images**

B.30 Medium scale - Scratchy touch images

Figure 67: **Medium scale - Scratchy touch images**

B.31 Medium scale - Smooth touch images

Figure 68: **Medium scale - Smooth touch images**

B.32 Medium scale - Velvety touch images

Figure 69: **Medium scale - Velvety touch images**